Elite, an Ethical 3rd Generation P2P Search Engine

Killian Levacher, B.A.I Computer Electronic Engineering

A dissertation submitted to the University of Dublin, Trinity College,

in partial fulfillment of the requirements for the Degree of

M.Sc. in Computer Science

University of Dublin, Trinity College

September 2007

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Killian Levacher

September 17, 2007

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Killian Levacher

September 17, 2007

Acknowledgments

I would like to take this opportunity first of all, to express my gratitude to Stephen Barrett for his kindness, positive advice and guidance, including all of those interesting and enjoyable conversations, very much appreciated, during the year.

I am also very grateful to Anthony Harrington's generosity, precious recommendations and time on probabilistic theory.

Of course, these last few months wouldn't have been such a challenging and rewarding year without our lecturers who were able to confer on us their enthusiasm and excitement in distributed technologies.

Additionally, these acknowledgments would be shallow if it wasn't for my beloved family, including Kate, who were so kind and supportive over the last five years.

Finally, and without any doubt, the most satisfying, fulfilling and enriching experience of all must be awarded to our very close cosmopolitan class. I sincerely wish our friendship will last for a very long time where ever we may be in this world.

Killian Levacher

University of Dublin, Trinity College
September 2007

Elite, an Ethical 3rd Generation P2P Search Engine

Killian Levacher, MSc NDS

University of Dublin, Trinity College, 2007

Supervisor: Stephen Barrett

Owing to the technological revolutions of the past century, information access and provision have become a fortiori, one of the most prominent corner stones of our society. Along with these revolutions, the associated responsibilities and choices of authorities and engineers designing and delivering these technologies to the community have increased tremendously.

Since its creation, the World Wide Web, for instance, has exceeded all expectations relating both to its size as well as the way it affects our lives. Intertwined with this exponential growth, the ability however to retrieve useful information from the net is now completly dependant on how search engines order and present these Web pages to users, raising many ethical implications.

Moreover, although the last decade witnessed giant leaps in information retrieval technologies such as Page Rank, important issues relating to the way search engines organise this data still persist. Traditional indexing approaches for instance, are finding it increasingly difficult to keep up with the internet's expansion, while link manipulation is moving Page Rank's hype curve towards increasing disillusionment. The recently observed "Rich get Richer" nature of the Web favouring high profile pages as well as entrenchment effects induced by search tools also pose a serious threat to diversity of content in cyberspace. Finally, concern among the general public over privacy and freedom of expression in the digital world is increasing as both goverments and the private sector are accumulating control of digital information.

The question "*Can search engine design be improved towards achieving better InfoEthical practices?*" evidently emerges.

This thesis therefore, is an attempt to provide solutions to these issues through the conception of *Elite*, a novel P2P Web search engine designed with InfoEthical considerations. *Elite*'s objective is to promote quality & diversity of content through selective collaborative peer filtering over a semantic overlay. A new *Ripple Effect* technique will also be designed and analysed with the aim of fragmenting and handing control over digital data to Web citizens through democratic decision making based on universal values.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Rise of InfoEthics

Owing to the technological revolutions of the past century, information access and provision have become a fortiori, one of the most prominent corner stones of our society. Its importance is such that it is considered as one of our most fundamental rights & duties, inherent to UNESCO's constitution [65] stating:

> *"That the wide diffusion of culture and the education of humanity for justice and liberty and peace are indispensable to the dignity of man and constitute a sacred duty which all the nations must fulfill in a spirit of mutual assistance and concern".*

Along with these revolutions, the associated responsibilities and choices of authorities and engineers designing and delivering these technologies to the community have increased tremendously. According to a UNESCO report [25], "*Because choices in their* ***design*** *and use carry moral consequences, these technologies pose significant infoethic challenges* ".

As this thesis is being submitted, the European Union is holding the third conference on the "Ethical Dimensions of the Information Society" in Strasbourg with the sole purpose of adopting a "Code of Ethics for the Information Society"[64] currently drafted and proclaiming an ethical framework including the following guidelines which will be referred to on several occasions throughout this thesis:

Extract of the Code of Ethics

- **Article 1.4:** This Code of Ethics [...] shall provide a common framework for the setting out of commitments to ethical conduct in the information society, in order to promote diversity of content in information networks.
- **Article 2.2:** The use of ICTs and content creation should respect human rights and fundamental freedoms of others, including personal privacy and the right to freedom of thought, conscience and religion in conformity with relevant international instruments.
- **Article 3.1:** Governments, civil society, the private sector and other stakeholders holding information on the state of technology should disseminate such information to the maximum extent possible to enable the public to consider how the information society can guard against use of information and communication media that violates human rights.
- **Article 5.2:** Principles urging the information society to be based on universally accepted values and all stakeholders to promote the common good and prevent the misuse of ICTs should be respected.

1.2 Search Engines in an Exponentially Growing World of Information

Since its creation, the World Wide Web has exceeded all expectations related both to the amount of information it can hold as well as the way it affects our lives. Several studies throughout the years have already attempted to estimate its actual size. According to [69] in 2003 the surface web[1] consisted of approximately 167 Tera Bytes of information. With a deep web[2] estimated to be 550 times larger, the total size

[1]The surface Web (or visible Web) refers to the part of the Internet directly indexable by conventional search engines

[2]The deep Web (or invisible Web) is the portion of the Internet not directly indexable by search engines It can be contained for example in databases. By definition, the deep web is the WWW which isn't part of the surface Web

of the system was evaluated to be more than 92 000 Tera Bytes ! Moreover, scientists are expecting this growth to persist in the future.

Victim of its own success, the web is now facing the challenge of avoiding collapse because of its own weight. Intertwined with this exponential growth, the ability to retrieve useful information from this colossal entity is becoming, without any surprise, a humongous task. Search engines have hence become the central figure in web browsing and the idea of managing without them today seems completely impractical. As pointed out by S. Olsen[49], "if your page isn't indexed by Google, your page doesn't exist on the web". *In a universe in constant expansion, our own vision of this world can only be seen through the eyes of these telescopes.* However large this source of information might be, our ability to use this data is now completely dependent on the pages presented to us by these search engines. Moreover, as most users only focus their attention on the top-ranked results [36, 47], the additional concern about the manner in which these pages are ranked becomes predominant. Furthermore, bringing these issues up to a broader context, it becomes crucial, for the reasons outlined in the previous section, to ask ourselves the following questions: Who controls this view presented to us of the web? Are the tools provided presenting a fair and reliable view of the web? Can they be improved? However genuinely good these authorities might be, should they possess such an amount of responsibility towards our society? Can they themselves be subjugated by anyone? And if so, who are these individuals? Are there any alternatives to this scenario? Can search engine design be improved towards achieving better Infoethical practices ?

1.3 Analysis of existent Search Engines

As Google puts it so famously well, "some people think the search is over" and why wouldn't they? The past decade, has witnessed giant leaps in information retrieval technologies. The most symbolic example could probably be described by the elaboration of this new ranking system, PageRank, [50]developed in the early 90s, by two Stanford students, Larry Page and Sergey Brin, suddenly improving the quality of our search engine queries and leading to the creation of, without any doubt, one of the most globally influential institutions of our time. Some of us might still remember the B.G. , or Before Google era where a random individual's home page would come

up on a search engine's first page result set, when you entered the word "search" as a keyword. Although this memory would make most of us smile today, it nevertheless illustrates how this ranking system has improved our ability to discover interesting information on the web. Fewer people however, would be aware of the difficulties, in the background, faced by search engine designers today, trying to adapt to the never ending evolution of this first Universal "Book" of Human Knowledge.

1.3.1 Copying the Web

As pointed out in 1.2, the web has become an immense entity growing exponentially, and simultaneously increasing the difficulties in both managing and organizing it. Until recently [12], the typical approach in dealing with this problem would have consisted of crawling the web using spiders to index and store as many pages as possible in servers. Without any surprise, this task has become Herculean and even today's most successful systems are struggling to keep up with the pace. According to a BBC report in 1999[20], search engines were already logging only 16% of the publicly indexable web estimated to be of only over a billion pages at that time[14, 42]. Another article[23] issued more recently reported that Google search crawlers could collect about 850TB of information from the web. Although this may seem a lot, it still represents only 1% of the estimated total size of the web 1.2. In addition to this, as mentioned by Junchoo Cho in [12], we also need to take into account the fact that existing pages are continuously updated. According to his study, 23% of pages change daily and pages have a half-life of only 10 days (in 10 days half of the pages taken into account were gone).

Notwithstanding automated crawling 10 years ago seemed the most obvious strategy to adopt, these figures strongly suggest these methods are becoming questionable. As pointed out by Faroo [2], "*We can't possibly copy the entire web to servers?* ". And after all, why should we? As stated in [15], is it really necessary to index the entire web? Although returning millions of pages to a query might seem impressive, is it really useful? As mentioned earlier, web users only browse through the first results sets anyway and then give up. Wouldn't pre-filtered indexing approaches be preferred to exhaustive indexes?

1.3.2 PageRank, or The Rise and Fall of an Empire

PageRank has undoubtedly become one of the most popular IR successes in the last few years, more than likely due to attention emanated from its resulting search engine. Originally described by Larry Page and Sergey Brin in 1998 [50], this algorithm exploits the linkage structure of the web to evaluate and sort out pages in relation to their "importance".

The intuition behind PageRank is the following: if a page possesses many links pointing towards it, we may conclude that a substantial amount of people are interested in this page. Additionally, if several people are interested in it, this page's quality should be considered as high. Furthermore, if one of those inward links originates from a "high quality" page, supplementary weight can be associated with this connection.

The simplicity of this algorithm and evidence of improvements over its predecessors, in result set ordering is today undeniable. Over the last few years however, scientists have started witnessing undesirable effects of this technique. Whether they are intentional or natural side effects, these occurrences have moved PageRank's Gartner hype curve peak[30] towards increasing disillusionment.

1.3.2.1 PageRank Manipulation

In recent years, extensive press coverage has pinpointed several incidents directly related to vulnerabilities with this technique. Although some individuals may benefit from it, to most of the Internet community, these events are becoming an increasing nuisance. Apprehension over the potential impact of these vulnerabilities related to the information presented to web users is rising.

In 2003, the BBC [19]described how president George W Bush had been Google bombed[17]. During the time this article was released, whenever the words "Miserable Failure" were entered in the search engine, web users were directed towards the president's biography on the White House website. During the same period, what started as a private joke by a pharmacist became an Internet hit. A site which was designed to look like an error message appeared among the first topped ranked result set[21]. When the query "weapons of mass destruction" was entered in the popular search engine, a page with the message: "These Weapons of Mass destruction cannot be displayed" appeared on the screen. Although the latter had very benign effects, these

Google bombs (also referred as Link bombs), which exploit the PageRank algorithm by artificially increasing the amount of links pointing towards a particular page, can also have serious impacts. Online ranking for example, can have important economic effects[20], hence web positioning companies such as NetBooster [18][4] have sprung up, offering services increasing your page rank in common search engines by misusing the algorithm for financial purposes. Even worse, last October The International Herald Tribune[63] reported the existence of a project among the US Republican party whose sole purpose was to increase the rank of 50 republican candidates to the presidentials by flooding the web with references to these politicians and repeatedly cross-linking specific articles on sites.

This series of events, among many more others, defeats the popular idea that search engines are unbiased and "untouchable" by malicious individuals wishing to influence popular opinion in a specific direction.

1.3.2.2 Rich get Richer Phenomenon

In addition to intended effects, PageRank also overlooks issues related to the "natural" structure of the web. A whole field of research has recently emerged, interested in the natural linkage of the World Wide Web. These studies have discovered how this hypertext document system evolved into a form very similar to a "bow tie" . According to [9], the distribution of outgoing and incoming links naturally follows a power law distribution favoring a core of the web corresponding to the knot in the bow-tie. This phenomenon has become so widely accepted as been part of the essence of the web, that all major analytical studies [13] modeling this data structure take into account this phenomenon. It produces a so called "rich get richer" scenario where existing pages with already a high degree of incoming links pointing to them are the most likely to receive new incoming links by newly created pages.

This characteristic once again, involves numerous ethical implications related to the dissemination and variety of information available to web users. To mention one among several studies[31], the political science report by Hindman in Princeton: *" 'Googlearchy': How a Few Heavily-Linked Sites Dominates Politics on the Web"*[37] illustrates how this "winner takes all", power law distribution aggregate structure of the Web is anti-egalitarian and exhorts global citizens to be aware that the

current predominant feature of online political information is based on what they call "Googlearchy", the rule of the the most heavily linked.

The natural tendency of this universal information source to favor high profiles inclines search tools to instinctively orientate themselves towards these pages and thus limiting the diversity of information accessible to web citizens consequently going against Article 1.4 in the Code of Ethics 1.1. Although, PageRank's original designers were, more than likely, unaware of this phenomenon, current engineers hold the responsibility to counter this natural effect while designing new web information retrieval tools in order to honor this code of ethics.

1.3.2.3 Entrenchment Effects

Another important fact needed to be taken into account, is how current search engines were designed based on a conceptualized static Web. For a surprising amount of scientists, considerations of the "*living*" nature of the World Wide Web is usually neglected if not entirely ignored. As mentioned previously1.3.2.2, the hyper link structure we created more than 20 years ago has evolved unintentionally with a bow-tie like anatomy and will continue to do so in the future. According to Liu, "temporal dimension of search is of great importance in the development of search technology"[51] and several studies[38, 55] have already started taking seriously into account this previously unexplored search dimension.

Emerged from this very young field of research is the concept of Entrenchment Effect. In several of his researches[40, 41, 56], Junchoo Cho studied the impact of current search engines on the Web and discovered how they delay the popularity of newly created pages by an alarming factor of 60 ![40] This literally means that *"if it took one year for a page to become popular without search engines, it may take up to 60 years for the same page to become popular when search engines are heavily used*". Unlike the one mentioned previously 1.3.2.2, this issue however, is directly related to the design of our search tools. As if a bow-tie knot 1.3.2.2 information structure favoring a core of the web wasn't enough, these tools have contributed to tightening the knot even more resulting in a theoretical *Web Choke* where randomly browsing information outside of this core entity would be extensively difficult or maybe even impossible. Not only does this problem contribute to the one mentioned earlier 1.3.2.2, it additionally impedes

citizens of their fundamental right to freedom of speech mentioned in Article 19 in the Universal Declaration of Human Rights[48]. One might argue that low profile citizens are still free to express their opinion on the net, but what benefit is there in speaking when no one or few are listening?

1.3.3 Freedom of Expression and Control over Digital Information

The original naive dream of a World Wide Web, ultimate cyberspace ensuring Global & Equal Freedom of Expression is fading as it blooms out of its infancy.

1.3.3.1 Digital Censorship

All of those thinking the Internet would finally become the decisive tool against censorship have been up for a big disappointment these last few years. History repeats itself, and censorship is now moving up to the global scale. Anyone reading the press recently would be aware of legal decisions and government actions attempting to control somehow the information flow over the net. At this instant, probably the most covered topic on digital censorship would involve the Chinese government defending its new Internet censorship laws ensuring " harmful information from spreading through the Internet" as described in the International Herald Tribune[62]. These actions are without any doubt impudently affecting global citizens from their fundamental right of freedom of expression [48]. For this reason, many organizations [58, 29, 67] have emerged, with the sole purpose of ensuring these rights are preserved. As engineers providing these technologies which can be used by some governments for the latter purposes, it is crucial to take into account these issues when designing these systems so as to minimize the potential harm induced by these activities.

1.3.3.2 Control of Digital Information by the Private Sector

When it comes to search engine technologies, the two obvious major concerns to be taken into account consist in minimizing the amount of influence a government can have over the result set delivered by the technology providers and ensuring the impartiality of the private sector creating these result sets. When it comes to the former, laws

can be created to guarantee the autonomy of search engine providers but as history shows us, these laws can be bent or even changed with time due to political shifts and hence these solution only provide a limited peace of mind for web users. As far as the latter is concerned, laws once again must provide a legal framework within which these technology providers must evolve, but once more, users are mostly relying on the good will of these entities. Unfortunatly, the "Don't be Evil" motto[32] isn't a view shared by all, and although some might currently embrace it fully, the considerable power generated by this amount of control over digital information is a too bigger bait for evil wills. The bottom line on all these issues is really all about who should possess this control over digital information? An attempt to answer this question is given in 1.3.3.5.

1.3.3.3 Anonymity

Anonymity of course, is also intrinsically intertwined with freedom of expression. A list of influential personalities could be enumerated, such as Voltaire, Camus, Hugo... who all, at some point in their lives, used anonymous writing in order to bring forward novel ideas which could have led to sanctions by authorities at the time. Anonymity thus, enables individuals to bring forward challenging ideas without having to fear for their own sake. This topic is therefore a crucial element needed to be taken into consideration when designing a technology promoting freedom of expression.

1.3.3.4 The Cons of Unilateral Freedom of Expression

Several endeavors have already aimed at tackling these issues. One of the most famous examples would be the FreeNet project[3] initiated by Ian Clarke. FreeNet is a distributed storage system built in such a way to guarantee total freedom of expression over the net. This example probably illustrates best how design considerations can have a major impact on Infoethical issues. Notwithstanding this project attempts a fair trial towards a solution to freedom of expression over the net, it nevertheless provides this freedom by stepping over other fundamental rights. According to FreeNet's philosophy[27], "there is no middle-ground, [...] you either have censorship, or you don't" and following this argument Clarke makes several conclusions such as "you cannot guarantee freedom of speech and enforce copyright law" and even more

concerning, in paragraph 6, he argues in favor of letting racist information spread freely over the net. How many other examples are needed? Child pornography for example isn't even mentioned as well as rights to privacy which, to the same extent as freedom of expression, is part of our most fundamental rights as well (Article 12[48] and 1.1). Should all these rights be neglected for the sake of total freedom of expression?

Although censorship should be avoided as much as possible, unilateral freedom of expression undermines other fundamental rights and therefore isn't defensible. The main reason behind this unilateral view emanates from his concept of control over digital information. According to him, "you can't allow those in power to impose "good" censorship, without also enabling them to impose "bad" censorship", hence to avoid any "bad censorship", Clarke advocates the total lack of control over information available to users within his system. Albeit any entity retaining the power to censor can indeed exercise it with both altruistic and malicious means, why should "those in power" as he calls them, or more precisely governments and the private sector, necessarily have such an authority?

1.3.3.5 E-Governance in Cyberspace

E-Governance[66] is an emerging concept which aims at "encouraging citizen participation in the decision-making process" as well as "making governments more accountable, transparent and effective". With recent advances in technology, it becomes increasingly possible to alter how citizens relate to and are governed by their government. Digital information management is, without any surprise, an area affected by this progress. These technologies, for the first time, offer the possibility for citizens to directly interact with decision making processes affecting them and therefore enables "the people" to increase their individual as well as collective power over their environment.

It has been previously shown 1.3.3 how the amount of power, over digital information, currently retained by authorities raises important Infoethical issues needed to be resolved. Section 1.3.3.4 additionally suggested how a certain amount of control over information is nevertheless necessary to protect the most vulnerable as well as our fundamental rights. E-Governance could provide an answer to these issues, by migrating and fragmenting this amount of power down to individual users, " Global

Citizens of the Web", promoting democratic decisions on how this Global Commons of Information is managed and used and by adopting Universal Values 1.1 shared by the majority. At the same time as collectively and directly empowering individuals to self governance over information management, this would release authorities from ethical implications related to the manipulation of data against the will of the people since the possibility of doing so disappears. Authorities such as the private sector would instead only be responsible for delivering these technologies enabling citizens to make collaborative decisions on how to manage the data available. Of course, this remaining responsibility still raises important ethical implications since it indirectly affects how decisions will be taken (whatever they are). This issue however, can be resolved by promoting openness in the design and implementation of these tools through open source applications for example. The weight of this bait to malicious entities, currently carried by the private sector, therefore disappears and is instead transferred to the population which is, (based on the fundamental concepts underlying democracy), a lot harder to subjugate as a whole.

1.4 Philosophies behind this Research

This Thesis believes that:

- Although censorship must be avoided, it shouldn't be done to the detriment of privacy and other fundamental rights for all the reasons mentioned above.
- The right to manage this Universal Commons[33] of Information should be possessed by citizens collectively as opposed to an institution or authority and decisions on how to manage this information should be done according to universal principles chosen democratically over the web. Notwithstanding this task may never be fully achieved, purely due to technical reasons, it nevertheless should be an ideal sought for by technology designers.
- The private sector should transfer the responsibility of controlling digital data down to the people, rather than holding the responsibility itself, and instead provide an open platform enabling individuals to interact and take these decisions together (1.1).

1.5 Aims of this Research

The aim of this research is therefore an attempt to provide a Search Engine as an initial application substrate designed with E-Governance purposes in mind, with an emphasis on removing as many Infoethical concerns as possible with the following characteristics. The system should:

1. Be Autonomous & Manage itself on a Global Scale.
2. Return High Quality Results to Web Users
3. Provide an Answer to the Increasing Size of the World Wide Web and its implications for Search Engines Performance.
4. Take into account the Evolution of the Web's Hypertext Structure.
5. Counter the Rich get Richer Nature of the Web.
6. Minimize Entrenchment Effects of New Web Pages.
7. Remove the Ability to Artificially Increase the Rank of a Page in Result Sets.
8. Remove Censorship going against Universal Values shared by the Global Community.
9. Promote Free Expression as well as Protecting Privacy and Fundamental Values.
10. Guarantee Anonymity to Web Users
11. Fragment and Disseminate the Power to Control Digital Information towards Global Web Citizens and Promote Web Democracy

1.6 Contributions of this Research

This research is an attempt to show how computer engineering design, with a strong ethical dimension, can have a direct and positive impact on how our society evolves, in the near future, through the design and implementation of *Elite* a new peer to peer search engine.

This document demonstrates how *Elite[43]* attempts to solve issues mentioned in this chapter. It shows how collaborative selective peer filtering for instance, can overcome the "Rich get Richer" phenomenon, while entrenchment effects can be reduced through the use of a zero input *Ripple Effect* recommendation algorithm developed for this purpose. A strong analysis of this algorithm will be presented suggesting it improves general Web browsing quality and has a major impact on new high quality pages awareness in the community. This research also shows how pre-filtered indexing scales in comparison to exhaustive indexing, to the never ending increasing size of the Web.

Finally, *Elite* ultimately endeavors to provide a reliable answer to control over digital information on the Web. It will show how a fair censorship model based on "Report & Support" combined with a "Tit for Tat" strategy can be achieved. *Elite* will eventually demonstrate how fragmenting its global index & associated responsibilities, by handing them down to users, can enable people together to decide how information should be managed by democratically promoting universal values.

1.7 Dissertation Outline

This research is organized as follows:

Chapter **??** will cover topics of interest to the reader which will be mentioned on several occasions within this document, followed by a detailed description of Elite's architecture and design in chapter 3.

The subsequent chapter will present how an analysis of the *Ripple Effect* was performed through formal computation and simulation. This chapter will also give a detailed implementation description of *Elite* from a software point of view.

Chapter 5 will present interpretations of the *Ripple Effect*'s formal model as well as simulation results and evaluation of *Elite's* design against the issues raised in this introduction.

Finally, a conclusion of this work will be given in chapter 6 followed by interesting post investigations to be carried out in future research.

Chapter 2

Related Research

This chapter presents an overview of previous research carried out which will be refered to on several occasions throughout this document. Section 2.1 outlines in more details the fundamental issues behind high profile favouritism while section 2.2 gives an overview of peer to peer technologies within a search retrieval context. Finally, section 2.3 contains a description of the underlying overlay network technologies used within *Elite.*

2.1 Rich get Richer & Entrenchment Effects

In several of his researches [40, 41, 56], Cho studied the evolutive nature of the World Wide Web and the impact of search engines on its ecology. One of the most recurrent subjects examined in this area of research relates to how high profile pages are favoured by the intrinsic nature of the Web. As it happens, the layout of all incoming and outgoing links from pages on the internet follows a power law distribution. What this means, is that pages already holding a large amount of incoming links, are also the most likely to get new ones in the future. In an environment, where the importance of a page is determined by search engines on how many incoming links are pointing to them 1.3.2, this subject becomes predominent. Hence, in one of his researches [40], Cho attempts to identify the scale of this behavior and subsequently examines the impact of search engines in such an environment.

2.1.1 Natural High Profile Favoritism

So as to determine the natural tendency of the net to favor high profile pages, a snapshot of the Web was collected repeatledly over a period of 7 months. Each of these snapshots consisted of about 5 million pages from 154 different websites. For each collection, a directed Web graph was created describing the linkage structure between all of these pages. In a Web graph, nodes represent pages and edges represent links. So an edge from node A to node B represents a link from page A to page B. For each of these pages, a PageRank value was computed (section 1.3.2). Once a PageRanks was associated to each page in each snapshot, pages's incoming links evolution was examined during this period of time. It was observed that 20% of pages with the highest incoming links obtained 70% of the new links after 7 months, while the remaining 60% of the pages obtained nearly no incoming links during that period. This shows how serious this high profile favoritism naturally caused by the World Wide Web is affecting low profile pages.

2.1.2 Search Engine Induced High Profile Favoritism

Awareness of the Web's natural inclination towards certain pages led to an investigation on the impact of search engines over this situation was subsequently carried out. The fundamental problem with search engines is that they present to users a set of most popular pages ranked in accordance to their "importance" which has a tendency in making these pages even more popular to the community. The following study therefore consisted in determining how much influence this behavior had over page popularity.

In order to examine this bias introduced by search engines, two models on how users discover new pages were outlined. The first consisted of the Random-surfer model where users discover new pages purely by surfing randomly on the Web and following links. This model's purpose is to capture the case when users are not influenced by search engines. A second model where, in contrast, users always start exploring the Web by going to a search engine was also introduced capturing this time a search dominant browsing model. By analysing and comparing the popularity evolution of pages within these two scenarios, the bias induced by search engines could hence be identified.

The random surfer model was defined according to 2 concepts which will be

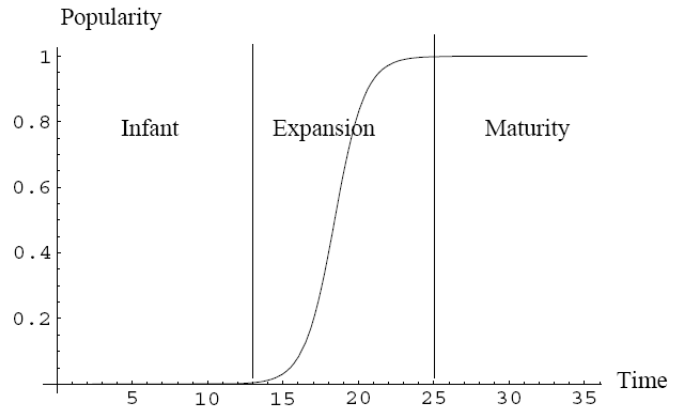

Figure 2.1: Time Evolution of a Page Popularity

refered to on numerous occasions within this document. The first important one relates to the popularity of a page. The popularity $P(p,t)$ of page p at time t is defined as the fraction of Web users liking the page. For instance, if say 100,000 users, out of a million, like page p, this page's popularity will be $P(p,t) = 0.1$. In a similar manner, the Visit rate $V(p,t)$ of a page p is defined as the fraction of visits a page gets within a time unit interval t. Using these 2 concepts, the following assumption is formulated: the number of visits to a page is proportional to its popularity. In other words,

$$V(p,t) = r * P(p,t) \tag{2.1}$$

where r is a normalisation constant. This assumption makes sense and is very obvious. If a page is more popular than another, we can very certainly expect the most popular page to receive more visits than the latter.

Based on these assumptions, a formal theoretical description of an average page's popularity is defined and graphed (figure 2.1).

An additional experiment based on the evolution of Googles popularity since it's creation was performed and confirmed this curve's accuracy. In this graph popularity $P(p,t)$ of a page of quality $Q(p) = 1$ (quality is measured from 0 to 1) is plotted against time. A quality equal to 1 represents the case where any user aware of the page will like it. From this graph, it is easy to notice how a page's popularity goes

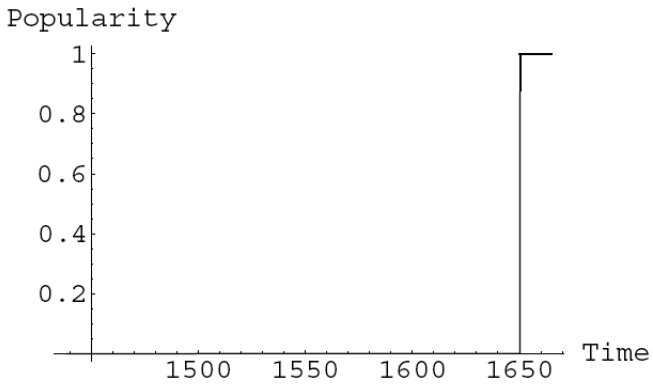

Figure 2.2: Popularity Evolution under the Search Dominant Model

through 3 phases: the infant stage, the expansion stage, and the maturity stage. In the first stage, the page is barely noticed until it reaches its expansion stage where it's popularity suddenly increases until it subsequently stabilizes at a certain value.

For the second model, it was assumed that all users where using the same search engine and that the same query was constantly initiated returning the same results. Moreover, in order to simulate user behavior faced with a result set, since it was proven that the probability of a page being hit by a user is related to the pages position in the ranking [36, 47], this behavior was modelled using the following equation:

$$V(p,t) = c * x^{\frac{-3}{2}} \tag{2.2}$$

where x is the rank position and c a constant. Figure 2.2 depicts the popularity evolution of a page within this search dominant model.

These figures are alarming ! For the same parameters, according to this graph, we can see how a page would take 66 times longer to become popular within a search dominant environment as opposed to a random-surfer model. Furthermore, an interesting point to note here is how the curve has now become a step function as opposed to a smooth evolution in the previous graph. This means that a page will possess a very low awareness initially for a very long period of time, however, as soon as it's popularity starts increasing, it stabilises nearly instantly.

These results therefore demonstrate how contemporary search engines have an

alarming influence on a page's ability to be seen on the Web, confirming our statement in section 1.2.

2.1.3 Page Quality Prediction

Having studied the popularity evolution of pages in these two different scenarios, Cho subsequently endeavors to predict page quality based on popularity evolution analysis. A formal definition of page quality is proposed: The quality of a page p, $Q(p)$ is the probability that an average user will like the page when it sees the page for the first time. In other words, $Q(p) = P(L_p|A_p)$where A_p represents the event that the user becomes aware of page p and L_p the event that it likes the page when seeing it for the first time. The problem with this definition is that it is very hard to determine if a user actually likes a page or not. And experiments using explicit feedback are very limited. Hence, he assumes that if a user creates a link to a page it must indicates that this user is interested in the page. Therefore, by observing the creation of links we can implicitly obtain feedback on how many people currently like the page. Secondly, Cho follows by stating that a page of high quality will see its popularity increase a lot faster than low quality pages since a larger fraction of visitors will like it when they see it. Ergo, by observing the increase in popularity of a page we can obtain an estimate of page quality as well.

The model described in the previous section is therefore extended to include awareness $A(p,t)$ simply defined as the fraction of users aware of page p at time t. This enables him to bring about another important relationship which will be used in a subsquent part of this document. In order for a user to like page p, it must be aware of it and like it. Therefore, $P(p,t) = A(p,t) * Q(p)$. Having extended this model, an experiment calculating the increase in popularity of pages over time $I(p,t)$ is initiated with the increase of popularity over time of a page mathematically defined as the derivative of this page's popularty, hence $I(p,t) = \frac{\frac{dP(p)}{dt}}{P(p)}$.

Figure 2.3, presents the results obtained for a page of quality 0.2. It is very clear, how the curve depicting the increase in popularity of a page represents the inverse of a page's popularity. Moreover, notice how $I(p,t)$'s initial value as well as$P(p,t)$ final value are equal to page p's quality. As a result of this study, Cho concludes by stating how page quality can be obtained by measuring it's popularity and increase in

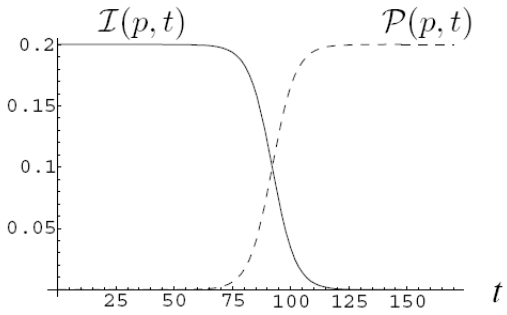

Figure 2.3: Time evolution of $I(p,t)$ and $P(p,t)$

popularity over time using the following formula:

$$Q(p) = I(p,t) + P(p,t) \tag{2.3}$$

2.1.4 3rd Generation Seach Engines

Taking into consideration the evolution of Web search engines in the last two decades, one could easily distinguish several "Search Engine Era".

The first generation of ranking systems for instance, determined the quality or importance of a page solely based on information contained in these. Very little consideration of the net's link structure was taken into account when ranking this information. We could probably estimate this period to have persisted until the end of the 20th century when came along PageRank.

The second search engine generation however, in addition to considering pages' information, started according much more importance to the link structure of the Web. Page Rank and the advent of Google for instance, in the early 21rst century probably symbolises best this evolution.

Today, Web scientists are considering going even further by taking into account, not only this global hyperlink structure, but also it's anatomy and evolution over time of pages it contains. This next step could characterise the birth of a new kind of search engine along with a new vision of this fantastic information common [33].

2.2 Peer to Peer Systems

A lot has been said on Peer to Peer systems these last few years, especially in relation to file sharing applications. These systems enable millions of user to connect with each other independently to collaborate so as to provide a common desirable service. This section describes in a nut shell current research developments in this area and potential features that, more than likely, will result from these investigations

2.2.1 Structured & Non Structured Peer to Peer

All of these systems are built on top of what is called an overlay network. Such a network enables peers to organize themselves through a set of protocols and rules. Most of these, can be divided in two categories referred to as structured and non structured overlays [24].

Unstructured overlays [6, 39] are built as random graphs made up of nodes linking with each other as they wish. These systems, due to their nature, usually require random walks or flooding throughout the network so as to discover data in the community. When a random walk occurs for example, each node forwarding the query evaluates it locally over the data it possesses and hands it over to one of its neighbor. They are widely used in popular applications because they can perform complex queries more efficiently than structured overlays. They do not impose any constraint on each node since the latter can choose whom they wish to be neighbors of. However, rare data is very difficult to find in these systems since information is randomly dispersed over the peers. Moreover, each query launched in the system will return different results based on the current state of the network.

Structured overlays [52, 54], on the other hand, assign keys to data items in the system and map each of these to corresponding nodes holding similar keys. Structured graphs are usually seen as being more expensive to maintain. However discovery of specific information in these systems can be more easily achieved due to constraints imposed on data storage and node placement. They enable exact queries to be performed in generally approximately $log(n)$ hops and return the same result set for each query initiated.

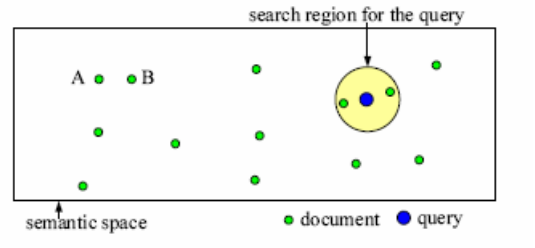

Figure 2.4: Psearch Semantic Overlay Example

2.2.2 Semantic Overlays

Semantic overlays belong to a level of abstraction above regular overlays mentioned previously. Their goal consists of organizing information semantically similar according to the underlying overlay environment chosen. Psearch for example [61], assigns vector space models to pages within its search space and places these items in function for their vector as shown in figure 2.4.

In this environment, the distance between 2 documents is proportional to their dissimilarity. When a query is initiated for example, it becomes as a result, very easy to orientate this message towards peers most likely to hold the information needed.

A wide range of semantic overlays [35, 60, 16] have been produced in recent years with very different and creative characteristics, organizing information in clusters or sets.

2.2.3 Local & Global Indexes

From these semantic peer to peer structures, applications were built [72, 71], including new search engines. Although very similar to traditional ones in appearance, their internal structure however is very different. One of the most important choices lies in choosing the correct index strategy to adopt which can be divided in 3 different categories: atomic, semi-atomic and non atomic peer indexing.

In atomic indexes, information model granularity is accomplished to a peer level.All of a peer index is grouped together and cannot be separated. The focus is made towards peers which can subsequently be modeled according to their center of interest contained in their corpus of information held. The advantage of these models is that peers can be grouped together by interests and keyword expansion can easily

be achieved. For example, if a query for keyword K_1 is initiated, and it happens that most of the peers interested in K_1 are usually also interested in K_2, an expansion that includes both keywords can be executed. The semantic implications of such a level of atomicity implies that knowledge of the individual dominates global knowledge. A certain amount of communal knowledge can only be obtained by joining several local corpus. In these scenarios, peers model pages they hold using vectors and these peers are then themselves modeled in the same way using their terms of expertise.

In semi atomic indexing, the granularity of atomicity is down to a peer's individual term index. Peer indexes can be separated in relation to the terms they contain but documents indexed can't. In these environments it is possible for example to determine which were the most popular documents for such a term, but term correlation for example becomes impossible (no query expansion possibility). Index storage in these systems are based on terms mapped with specific peers.

Finally, non atomic indexes possess a granularity at the URL level. The focus here relies more on the documents themselves being indexed. No peer identity is retained neither with terms or document correlations. Global knowledge here dominates completely. All terms present in a document are incorporated in a vector which is then used to locate it later on (similar to Psearch 2.2.2).

2.3 Pastry

2.3.1 Pastry Overview

Pastry [52] is a generic,structured, scalable middleware which acts as a substrate for peer to peer applications. It enables peers to form a decentralized, self organizing and fault-tolerant overlay on the Internet. Pastry's main objective is to provide efficient routing by organizing peers in deterministic locations. It is also a very convenient overlay for peer to peer applications since it provides support for object replication, and fault recovery.

Peers within a pastry network are arranged in a ring and given a unique nodeId in a circular 128bit space. Each peer maintains a routing table with peer Ids it currently knows about in the system. When a Peer receives a message with a 128 key, it routes the message to one of the nodes in its routing table holding the closest Id. Each peer

receiving such a message forwards it to the next node it knows about with an Id closer than its own until the message arrives to the node holding the closest Id in the system. This has the nice property of always guaranteeing that a message sent will arrive at a destination. The expected average number of hops for a each message is equal to $log(n)$where n is the number of peers in the system.

Furthermore, each peer in the system keeps track of L of its immediate neighbor peers (Leaf Peers) in its node space Id. This enables replication to occur as well as failure notification etc...

2.3.2 Scribe

Many middleware applications have been built on top of pastry. Among the most popular, Splitstream [45] enables high-bandwidth content distribution in cooperative environments while POST [7] provides an all in one communication system including mail, instant messaging etc... Also, file storage middleware have sprung up such as PAST [11, 53] maintaining a generic storage system for users and Squirrel [57] enabling peers to organize themselves and create a distributed web cache. A new global generic overlay [46] has even emerged just recently based on Pastry with the aim of linking together all kinds of overlay networks such as Chord [34], Tapestry [22], CAN etc... in order to provide them with a global bootstrap solution.

Although these application's popularity are increasing very quick, probably the most adopted of all would be Scribe [10, 44]. Scribe is a generic, scalable group communication and event notification system providing application level multicast and anycast. In a nut shell, Scribe enables nodes to create or subscribe to topics. Whenever a peer (Peer A) wishes to subscribe to topic T for example, it creates a 128 bit hash (H(T)) of this word and uses it as a destination Id in a Pastry message. The node receiving such a message becomes the root topic and maintains a list of child subscribers. If another subscribe message for topic T is forwarded within the network by peer A for instance, the latter stops forwarding the message and adds the source Id as its own child subscriber. This creates a tree of subscribers in the Pastry overlay. Whenever any peer wishes to publish information on topic T, it simply creates a message with this information and sends it using H(T) as a destination Id. The root then simply sends this information to all of its child peers in the tree.

Chapter 3

Elite Design

This chapter describes in detail *Elite*'s design, and in particular how it attempts to remove or reduce fundamental info ethical issues outlined in chapter *1*. It starts by stating the fundamental concept underlying this system in section 3.1 followed in 3.2 by an outline of the fundamental technologies chosen and their reasons. Section 3.3,3.4, 3.5 and 3.6 provide a deep insight into *Elite*'s internal operation. Particular attention should be given to section 3.7 describing the *Ripple Effect* technique as it will directly relate to the following Chapter. Finally, future possible design considerations are proposed so as to improve on this prototype's current design in section 3.8.

3.1 Intuition behind *Elite*

The fundamental intuition over which Elite relies on is the following:

> *"Human Beings are generally interested, to various degrees, in particular topics. Additionally people, most of the time, only browse over pages of interest to them. Hence, a page regularly visited must possess a certain quality to this user in relation to a given topic. Furthermore, if this page happens to be frequently browsed by a set of individuals highly interested in the same topic, this page must be of high quality in relation to this particular topic."*

Hence these individuals highly interested in a particular topic, referred to as "Elite" members, possess as a group, valuable information regarding the quality of

pages for this topic. This system is therefore designed so that the *"Elite"* can *Shine and Share it's Wisdom to the rest of the Community*" in exchange of some reward, in a similar way described by Wassily Kandinsky in *"Point and Line to Plane"* [70] where a small group in our society can positively influence and improve the majority by spreading new valuable ideas and information to the rest of the community.

3.2 Fundamental Design Decisions

The first fundamental decision taken when designing *Elite* involved using an underlying peer to peer network overlay. Such a technology would enable us to create an autonomous, decentralised system difficult to attack, manipulate and would also remove the need for any external authorities. Moreover, the overall load of the system could be shared by every peer as a whole, thus improving scalability. A structured overlay was chosen as it would enable us to organize nodes according to the semantic needs of the network and also provide a consistent answer to successive queries. Pastry was ultimatly chosen as Elite's underlying overlay since it is a mature and very good peer to peer middleware abstraction and also provides fault tolerant behavior, replication etc... which would later on be crucial choices for full scale deployment.

It was decided to use a pre-filtered index as opposed to traditional exhaustive indexes. As stated in section *1.3.1*, millions of pages returned in result set are never browsed. People usually take into account only the first few results and then start a new query or give up[36, 47]. So using a pre-filtered index would make *Elite* more scalable.

"Human Web crawling" will be used instead of spider Web crawling. Millions of people browse the Web simultaneously 24/24 and don't consume any cpu cycles. Moreover, as stated previously, people generally browse through interesting pages, hence using humans would both reduce computational cost and provide a first level of Web filtering which web crawling spiders can't achieve.

A hybrid indexing scheme will be used attempting to take the benefits of both local and global indexing. Since our system will be using peer selection mechanisms, it is mandatory to possess some level of peer modeling, hence a local indexing scheme enabling user modelling is required. However, our goal is to provide equal provision of information to all users, therefore a query should return the exact same result set to

any peer in the system where ever they might be located, ergo global indexing should be used to store our results.

Elite will attempt to counter Entrenchment effects by using Cho's quality prediction equations *2.1* along with a new *Ripple Effect* technique. Hence popularity as well as increase in popularity of pages in the system will be monitored.

A variation of this *Ripple Effect* algorithm will also be used to endeavor a level of censorship model enabling peers to democratically make decisions about how information is managed in the system.

3.3 *Elite* Overview

3.3.1 *Elite* World

In the "Elite World", users are modeled as peers surfing the Internet via a web browser. They maintain a corpus of pages previously visited, each associated with a given web page Id representing subjects of interest related to it. The corpus needn't store the actual web pages themselves, solely their URLs and web page Ids. The joined set of pages thus represents a pre-filtered portion of the Web "distilled" according to community browsing activities. Peers are subsequently modeled based on their web page Id set along with their browsing patterns and given an Id describing their interests and associated strengths. The system thereafter selects a group of "Elite Peers" of variable size and membership, comprising of individuals with highest interest in specific subjects. Elite Peer Visit Rates (VR) as well as Relative Visit Increase (RVI), for each web page co-jointly browsed by the group, are measured and shared throughout the entire community. Additionally, in exchange in offering this information, Elite Peers possess the privilege of being the foremost aware of newly created pages with potentially high quality in their subject of expertise. Finally, Elite resembles most search engine as to initiating keyword queries but differs in the result set retrieved from it. Web pages returned are ranked in 3 different ways, based on Visit Popularity, Increase in Visit Rate and Rippling (explained in further details in section 3.7).

3.4 *Elite* Architecture

3.4.1 Peer modeling

Peer modeling is a very large research field of its own which would probably require a dedicated chapter in this document. Since this research focused more on how peers could collaborate between each other to meet our goals, these peer in our system, were modelled in a simple similar way as achieved in [35]. Pages browsed are given an identity using Vector Space Modelling, which creates an array holding as entries, topics covered in these pages, with their associated strengths. Once any peer possessed a set of these vectors, the normalised average strengths in each of these terms was computed giving a global vector describing this individual.

It must be emphasised how this elementary peer representation does not affect our system since potentially any user modelling technique could be "plugged" in *Elite* as desired. However, a long list of variables could be taken into consideration for such a task. Note that in our system for instance, since the normalised average is taken for each term, this technique doesn't make any difference between the amount of pages browsed between users but only monitors the relative proportion of pages browsed for each term. A more evident approach would consist in evaluating a function taking into account for example, the number of pages browsed in particular topics, the average page browsing rate, the time spent on pages etc...

3.4.2 *Elite* Group Membership Management

For each individual keyword queried by users, an elite group is created based on identities obtained through peer modeling. This group's sole purpose is to deliver a set of pages of increasing quality related to this keyword.

3.4.2.1 Keyword Elite Peers

The Keyword Elite Peer (KEP), represents and acts as the main rendez-vous point for a given elite group. Any peer in the system possessing an Id which happens to match or closely resembles this keyword's hash is selected as the KEP. A peer can act separately as the KEP for several keywords if it happens to correlate with several hashes as shown

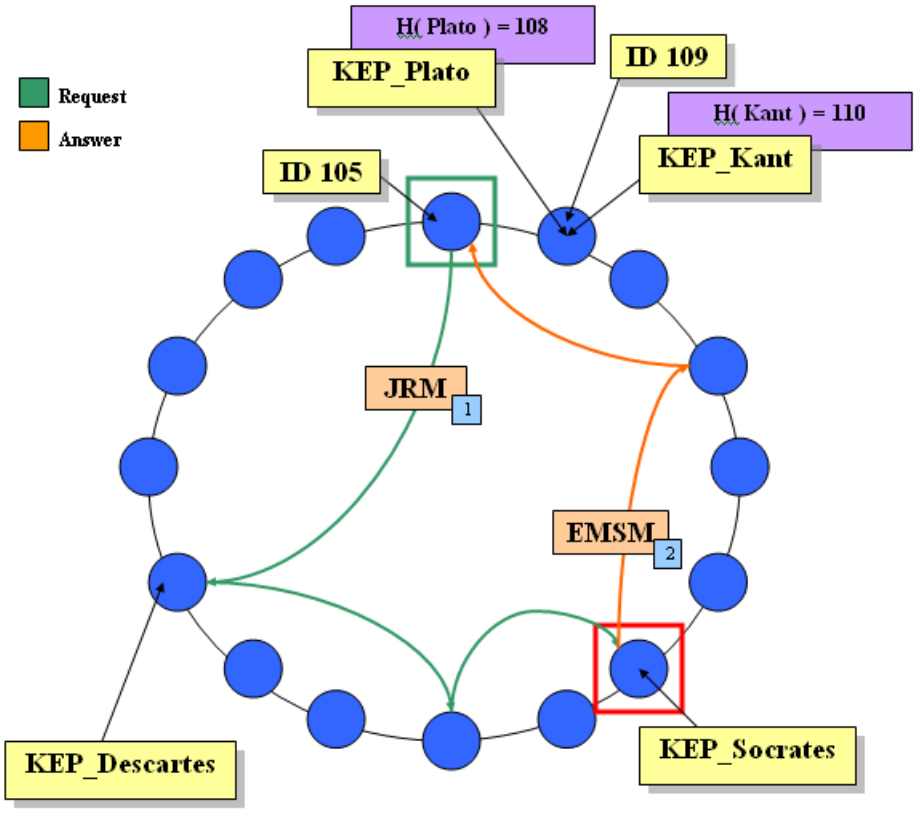

Figure 3.1: Peer Joining an Elite Group

in figure 3.1 but there can only be a single KEP for each elite group. It is responsible for managing the elite group's membership requirements and size according to the networks characteristics and popularity of it's associated keyword. In other words, it aims at maintaining the elite group's size at a stable number as high as possible to guarantee the best quality results to users as well as balancing this aspect with the networks bandwidth capacity.

3.4.2.2 Diagram Clarifications

Although the system consists of one ring it is more easily conceptualized as one global ring and several elite rings for each keywords 3.2. In the rest of this document, "Simple Peer" to "Simple Peer" messaging will be depicted in the global ring while Elite Peer

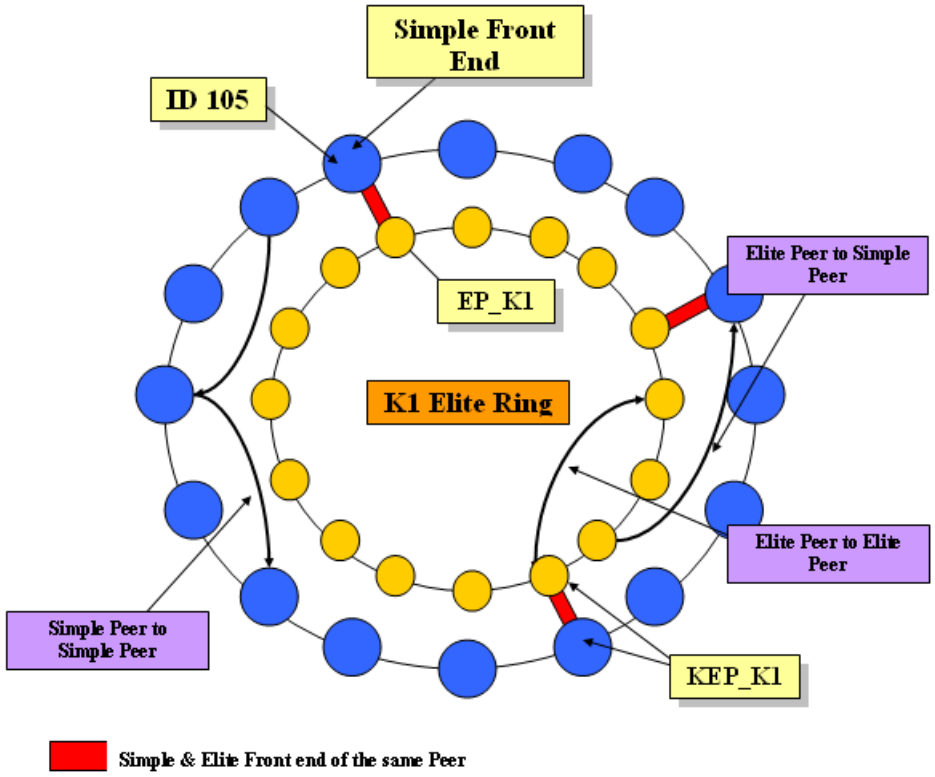

Figure 3.2: Elite Diagram Clarifications

to Elite Peer communication will be represented in the internal ring. Two nodes linked together simply illustrates how both refer to the same Peer but from different role perspectives. A message initiated from an Elite Peer and routed through the global ring thus describes how the communication was initiated by the Peer because of it's Elite status in a keyword but destined to a Peer which isn't necessarily an Elite Peer.

3.4.2.3 Membership Request

When a Peer browses through a web page containing a new subject of interest previously unknown (a new term), it creates a record Id of this new web page in its corpus as usual and therefore becomes initially "interested" to a very limited extent in this subject in proportion to the importance of this term in the Web page. When this happens,

this peer wishes to know if it's interest or expertise in this term meets the minimum requirements to enter the associated elite group (this initial situation will only occur when Elite is been initialized or if the term is very uncommon). It therefore sends a Join Request Message (JRM) to the corresponding KEP by setting this message's destination Id as the hash of the specified term 3.1. The latter sends back an Elite Member Status Message (EMSM) either granting the request, (if the current Elite Peer membership size for this keyword is below the maximum authorized or the peer meets the minimum requirements), or else refuses the request. When the second situation arises, the KEP sends back the current minimum entry requirements for this elite group. The Peer stores this information for each keyword it currently isn't an Elite member of, and initiates a JRM if it ever meets this expertise. Since we assume the overall expertise of peers in a given topic can only increase as they browse the web, this minimum requirement can only increase as well. Hence if a peer never meets this minimum entry requirements it never needs to send such a request again therefore avoiding unnecessary messages being sent.

3.4.2.4 The LEP Update Protocol

When a Peer is granted Elite membership, it subscribes to the Scribe STopic corresponding to the hash of this keyword. This enables all Elite Peers (EP) as well as the KEP to contact all current EPs without any knowledge of their identity thanks to Scribe's implementation. The Least Expert Peer (LEP) of the group periodically broadcasts on this topic it's expertise in the given keyword as it increases over time. The KEP which also subscribes to this topic is therefore constantly aware of the minimum entry requirements for a given elite group. Hence if it receives a JRM with an expertise higher than the current Least Expert Peer, it grants access to this new Peer and asks the LEP to leave the group by broadcasting the notification on to the group topic. When a new EP joins, it automatically sets itself as the LEP and broadcasts its own keyword expertise to ensure it is or isn't the LEP. Each time an LEP update is broadcast in the group, each EP compares it's own expertise level with the one contained in the message. If an EP happens to have a lower expertise than the current LEP, it now becomes the least expert of the group. It therefore sets itself as the LEP and starts sending periodic updates of its own expertise. The former LEP receives this update and

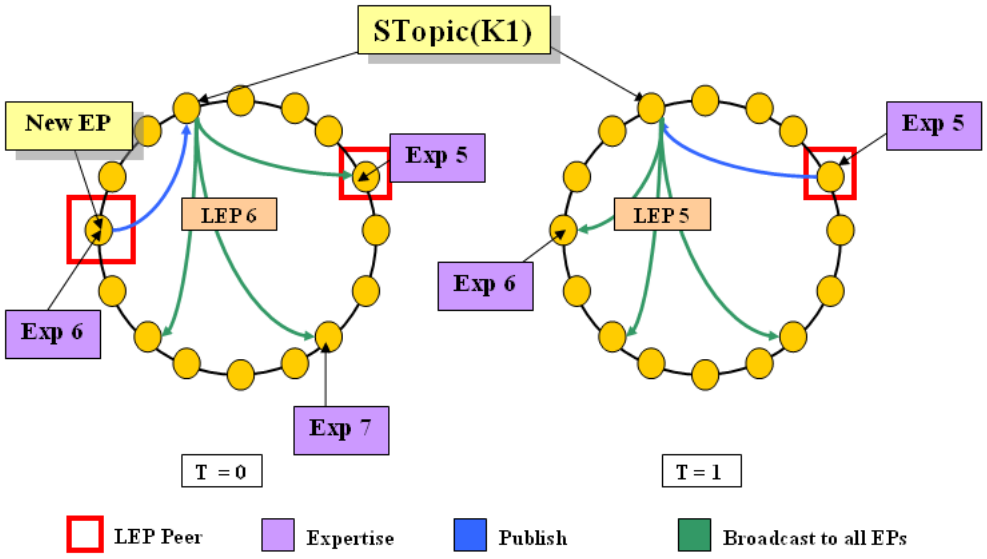

Figure 3.3: LEP Protocol

realizes it isn't the LEP anymore and hence stops sending updates as shown in figure 3.3. This protocol ensures that at any time, the LEP status is constantly assigned to at least one Peer in the group without any identity being revealed either to EPs or the KEP.

3.4.2.5 LEP Relief Task

If the KEP doesn't receive an LEP update after a certain amount of time, it assumes the least expert peer has failed and sends an LEP relief task. This consists of anycasting a message to a random EP asking it to initiate an LEP update. If this EP isn't the LEP (which it more likely isn't) then all other EPs with lower expertise will set themselves as LEPs and start sending updates. At some stage the EP with the least expertise in the group will have sent its update, thus informing all other EPs that they aren't the LEP anymore and the situation goes back to normal with a single LEP broadcasting its expertise as shown in figure 3.4.

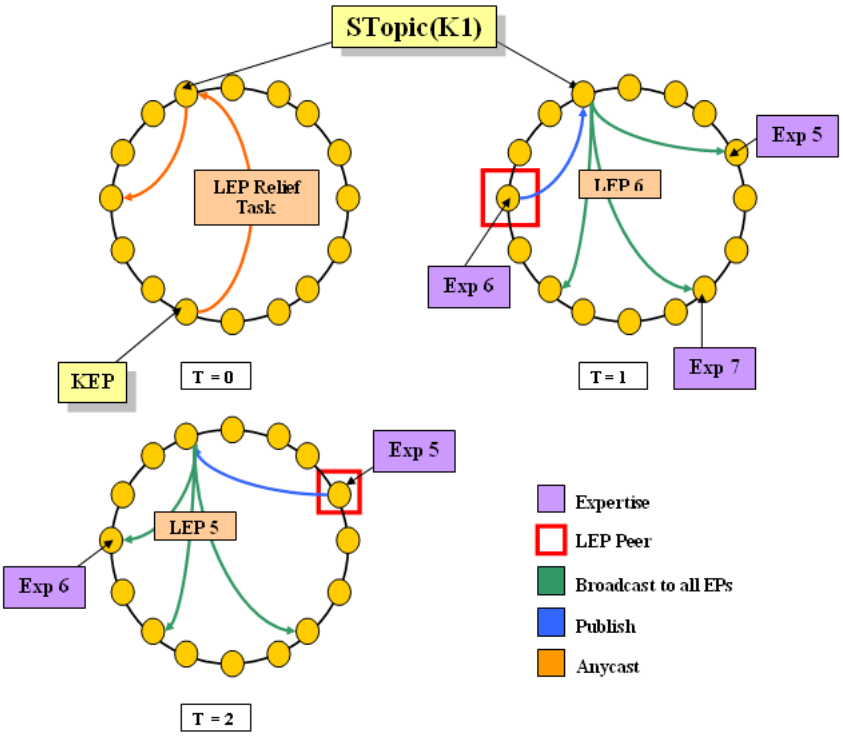

Figure 3.4: LEP Relief Task

3.4.2.6 Anonymous Elite Membership

Membership to an Elite group is totally anonymous even to the KEP. The latter only knows about the number of EPs and the current minimum entry requirements for the group. When a Peer receives a EMSM and is granted membership, it is given a temporary Elite Peer Id by the KEP. Each EP periodically sends an "Elite Peer Alive Message" (EPAM) with it's Elite Peer Id to the KEP. If the latter fails to receive this update in a given time it assumes the peer is dead and decrements the total number of EPs in the group. This ensures total anonymity in the system.

3.4.2.7 KEP Replication

In order to minimize repercussion of a KEP potentially failing (figure 3.5), the former continuously forwards information it receives related to the elite group size and current temporary Elite Ids to its Pastry Leaf Set Nodes (it's neighbors). These nodes also subscribe to the elite Scribe Topic (section 3.4.2.4) so as to receive minimum entry requirements for this group. They act as "dead ends" receiving information but never taking any actions and are for this reason, referred to as Latent KEPs. If at some stage the active KEP dies, the Latent KEP with the closest Id to the elite keyword's hash will ultimately receive a message destined for a KEP. When this happens, it sets itself as active and sets its own Pastry Leaf Set Nodes as Latent KEPs (most of them will already be latent KEPs).

3.4.3 Three Dimension Ranking

As mentioned in section 3.3.1, *Elite* provides an unusual result set to user queries. It consists of 3 different ranking systems each maintained for specific purposes.

3.4.3.1 Visit Rate Ranking

The first ranking, is the most straight forward. It consist of pages ordered according to their current visit rate. As illustrated by Cho [40, 41] (section 2.1), the quality $Q(p)$ of a page is related to its popularity $P(p,t)$ according to equation $P(p,t) = A(p,t) * Q(p)$ ($A(p,t)$, the awareness of page p during time t) while its visit rate $V(p,t)$ is directly proportional to it's popularity shown in equation $V(p,t) = r * P(p,t)$ (r a constant),

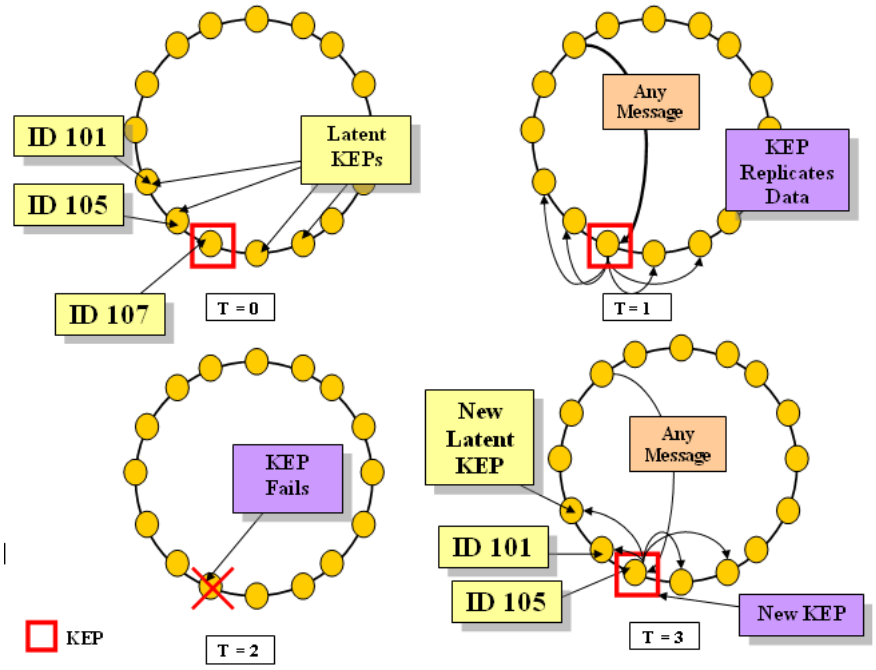

Figure 3.5: Latent KEPs

hence the visit rate of a page is dependent on it's quality according to the following equation:

$$V(p,t) = r * A(p,t) * Q(p) \tag{3.1}$$

. Measuring the visit rate of a page thus, gives a fair idea of it's quality. Furthermore, as discussed in section 3.2 *Elite* is designed with a hybrid architecture valuing Local information, or information possessed by individual peers. Hence, if we consider equation **??** from a local perspective, the awareness variable $A(p,t)$ becomes a Boolean since one Peer can only be aware or not of a page. When this peer becomes aware of a page equation **??** therefore simplifies to $V_L(p,t) = r * Q(p)$ highlighting the evident proportionality of a page's visit rate to its quality. The bottom line behind this fact is that this ranking is user driven as opposed to link driven as in Page Rank. This result set therefore counters the high profile favoritism (section 2.1) induced by estimating quality based on the linkage structure of the web. We have previously seen (1.3.2.2) how the web's structure naturally follows a power law distribution favoring high profile pages (pages with many inward links) and how this situation favors the "Rule of the

most heavily Linked"[37] phenomenon with every ethical implications associated with it. *Elite* however relies on a "Natural" rather than an "Artificial" Quality estimator (Human Beings rather than Web Links). Users implicitly assign a certain quality to pages they browse through their visit rate. Of course, a page's quality can have different values for different subjects but as stated in section 3.1, if this page's quality is considered as high by peers highly interested in the same subject we can confidently estimate this page's quality for this given subject as being high.

3.4.3.2 Relative Visit Increase Ranking

Although the first ranking mentioned previously counters high profile favoritism, it still fails to resist the "Rich get Richer" phenomenon. In other words, a search engine based solely on the first ranking will have the same effect than those portrayed in [40]. Since users usually only browse through the first result set[36, 47] Pages already highly ranked will get more awareness and hence will be visited even more. For this reason, the Relative Visit Increase (RVI) of pages is taken into account. As described by figure 2.3 in chapter 2, the relative popularity increase of a page, characterized by equation

$$I(p,t) = (\frac{n}{r})\frac{\frac{dP(p,t)}{dt}}{P(p,t)} \tag{3.2}$$

(with $I(p,t)$ is the increase in visit rate of page p during time t and n & r constants), is a good estimate for its ultimate popularity and hence its quality. Moreover since visit rate is directly proportional to popularity with some constant r (equation $V(p,t) = r*P(p,t)$ or $P(p,t) = \frac{V(p,t)}{r}$), taking the constants outside the derivative in **??** gives $I(p,t) = (\frac{n}{r})\frac{\frac{dV(p,t)}{dt}}{V(p,t)}$. Hence, measurement of a page's Relative Visit Increase value gives a fair idea of its quality at early stages of its life time. Consequently, the second ranking offers users results which aren't currently highly visited but possess the highest increase in visit rate by experts in this subject. Assuming the equation governing the probability of a page being hit according to it's ranking position in [36, 47] holds and that pages with the same rank on both list are hit with equal probability, the second rank list will have the effect of decreasing the rich get richer phenomenon. This rank list thus takes into account the evolution of pages through time and will have a tendency of shifting Cho's Popularity curve (figure 2.1) to the left.

3.4.3.3 Rippled Ranking

The third result set, is only retrieved by Elite Peers of a particular keyword query. Ergo, only peers issuing queries containing keywords for which they have the Elite Peer status will receive this result set, any other will solely get 2 result sets. This list consequently is perceived as a privilege for EPs in exchange for sharing their expertise to the community. This list consists of new pages this peer is currently unaware of which have been assigned a certain quality by other EPs. They are ranked based on the number of recommendations received by other Elite Peer members of the group, in other words the more a page is recommended the more it goes up the ranking. If one of these pages, is browsed by the user, the former becomes aware of it and the page is removed from the ranking. The purpose of this third ranking is to impede as much as possible entrenchment effects (section 2.1) of new pages with high quality. A more detailed explanation is given in section 3.7. Briefly, if a page of high quality for a subject is created and browsed by only one EP for this given subject, it will be spread around the Elite group with a speed proportional to its quality. Therefore, this system allows the awareness of new high quality pages to increase significantly within the elite group and thus increase their visit rate ranking.

3.4.4 Global Index Generation

Elite's design is based on a novel Global Keyword Distributed Indexing scheme (GKDIS) approach. It is global in the sense that a query initiated by any peer in the system will return the same result set, guaranteeing consistency in the quality of pages received by every individual, and keyword distributed in the sense that no single peer possesses the complete inverted index for a given keyword in contrast with other distributed search engines. This guarantees distribution of control over information in the system. If a peer for example, could somehow adverse *Elite's* security design and manipulate the information it holds, even then, it could only manipulate a small part of the index and thus have a limited impact on users.

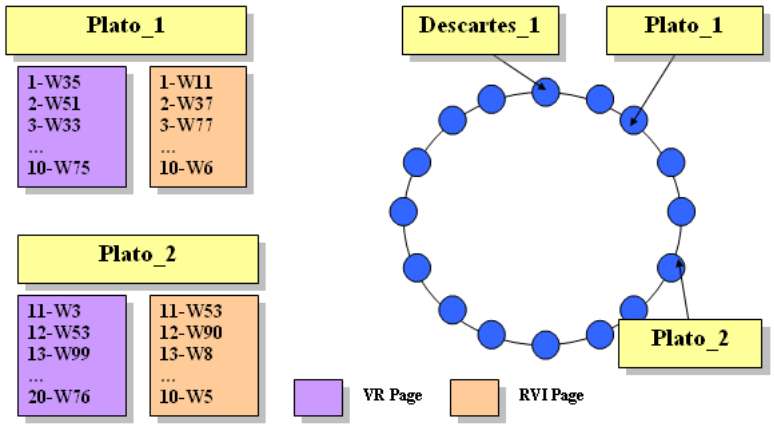

Figure 3.6: Page Elite Peers

3.4.4.1 Page Elite Peer

Elite stores and fragments its keyword index in several pages corresponding to each page retrieved during a query. A page thus holds exactly and no more than 10*2 entries: 10 VR ranking entries and 10 RVI ranking entries (exception for the number of entries are made and explained in section 3.4.4.4). They are named according to the following convention $<K_m>\_n$ (where n is the page number retrieved by the user and $n > 0$ & $n = 1$ by default). Each page's name is hashed and stored in the peer holding an Id closest to this hash (figure 3.6). Peers storing a copy of these pages are called Page Elite Peers (PEPs). Each one of them is responsible for keeping VR and RVI values up to date for each URL entry. The number of PEPs in the system and thus, the number of result set pages available per keyword directly depends on the capacity of the P2P network to hold these pages for scalability purposes. Hence the bigger the community, the larger the result sets. Each PEP subscribes to the <keyword> SCRIBE topic and the creation of a new PEP is broadcast to this topic making all EPs and PEPs aware of it's existence.

3.4.4.2 PEP Mapping

Every Web page ranked in the system can only be at one place at a time in each ranking. Hence Web page W1 for example, is located in one particular PEP page for it's associated VR value and another PEP for it's RVI value. Let's take into consideration

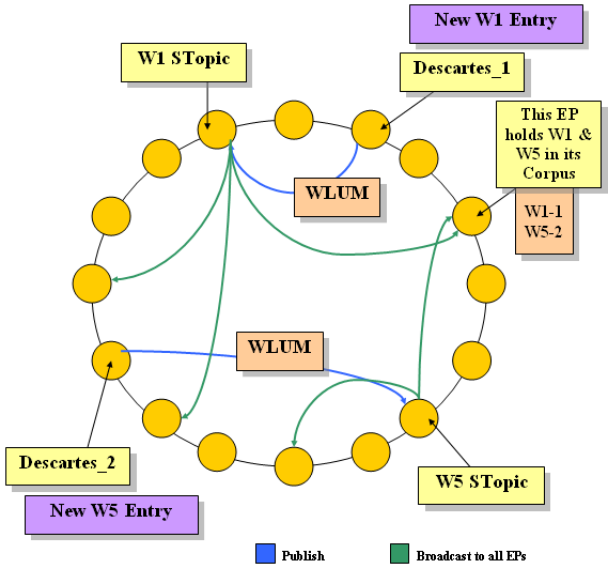

Figure 3.7: PEP Mapping

for the moment only VR ranking. For each Web page related to the keyword in which a peer possesses Elite status, a table mapping the PEP currently holding this page is created in this EP. This table needn't be complete, as this information is only needed if the EP revisits these pages as will be discussed in the following section. For each of these Web pages, every EP subscribes to a Scribe topic named with the following convention: "<keyword>_VR_<URL>" where keyword is the keyword of expertise and URL the Web pages address. Any information regarding this particular Web page will be broadcasted on this topic and hence every EP holding this Web page in their corpus will be notified. When a PEP inserts a new Web page entry in it's page it broadcasts a Web page Location Update Message (WLUM) to this Web pages topic, specifying this particular Web page's new location (figure 3.7). Every EP subscribed to this topic subsequently update their mapping table.

If an EP needs a PEP mapping for one of it's particular Web pages but hasn't receive a PEP update yet, it anycasts a Web page Location Request Message (WLRM) to all other EPs subscribed to this topic and waits for one of them to broadcast this page's current location. If no EP replies it assumes this is a new global entry in the system and maps this Web page with the latest PEP created (PEP with the highest "n" value) therefore placing the Web page at the end of the ranking. This

mapping done by EPs for Web pages PEP location on the VR ranking is similarly done by PEPs themselves for the 10 VR entries they possess and the corresponding PEP's holding these same entries on the RVI ranking. In other words, EPs and PEPs hold PEP mappings for each Web page they possess respectively for VR and RVI ranking. We therefore have a sort of double mapping layer. The corresponding Scribe topic for these PEP Web pages RVI ranking location follows the convention "<*keyword*>_RVI_<*URL*>".

3.4.4.3 Visit Rate & Relative Visit Increase Updates

Whenever a Peer which happens to be an Elite Peer for a given keyword visits a page related to it, it must inform the community (figure 3.8). To do so it finds the correct PEP holding this web page's entry by looking through it's mapping table described in the previous section 3.4.4.2, hashes the PEP's name and uses this hash as the destination source. Pastry routes the update to the correct Peer and the latter updates the global visit rate value it holds for this entry. This can result in the web pages rank to be increase or not depending on whether it surpassed the VR value of the entry above it. EPs send single VR updates as soon as they visit a Web page or VR updates bundles at periodic intervals depending on the network's capacity to handle this communication.

Additionally, PEPs measure the rate of change of their entries visit rate and at regular intervals, inform the proper PEP holding these entries RVI values of this change. It finds the correct destination in exactly the same manner as EPs did previously and sends an RVI update to another PEP.

3.4.4.4 PEP Entries Transfer

Distributing the index over several Peers does fragment control over an entire index but it nevertheless creates complications. Inconsistencies in ranking can arise between page edges. More concretely, if PEP "K1_VR_2" for example receives a VR update for it's locally highest entry, how does it ensure that this entry should still be ranked lower than PEP "K1_VR_1" lowest entry? To handle this issue, whenever a PEP receives a VR update for its lowest entry, it broadcasts a Lowest PEP Value Message (LPVM), containing its own name and new lowest value, on the Scribe STopic <Keyword>.

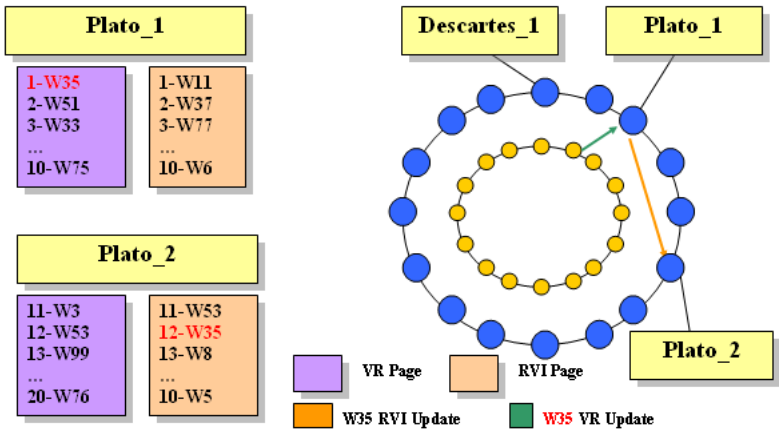

Figure 3.8: VR & RVI Updates

Every PEP keeps track of others PEP lowest value in a second table (with "$n-1$" rows one for every PEP but itself), hence when PEP "K1_VR_2" receives an update for its highest entry, it checks whether or not this entry should be instead in another PEPs list. If this is the case, it sends a Transfer VR Web page Message (TVWM) to this PEP containing the entry and it's associated VR value. At this stage, the PEP sender holds 9 entries while the receiver now holds 11 VR entries which causes an overflow. When an overflow occurs a PEP of page number $n$selects its lowest value (among the 11) and transfers it to the next PEP of number $n+1$. This re-stabilizes the system to PEPs holding both 10 VR entries again. If PEP number n in a network containing n Page Elite Peers contains 10 entries and happens to receive an additional one, (this can happen if a new Web page is added to the system), it overflows and transfers it's lowest entry to a new created PEP with number (n+1). When this happens, this new Page Elite Peer only holds 1 entry. This is the only time a PEP can hold a lower number of entries than specified in section 3.4.4.1. These page transfers, of course also apply for RVI entries.

3.4.4.5 Web Scalability

Finally, as our goal isn't to index the entire web as stated in 3.2 for reasons discussed earlier (1.3.1), but instead only to keep in index useful pages, a "garbage removal" function can be used by the last PEP (PEP with highest n value) to remove pages with a very long life time and low visit rate. Only this PEP can use this functionality

since it holds the entries with the lowest VR & RVI values. This simply means that in order to survive in the system, a web page must possess a minimum VR or RVI value above an arbitrary threshold. This enables the system to scale according to it's network capacities by preventing pages with low quality to clutter the system.

3.5 Elite Query

3.5.1 Single Keyword Queries

Once Page Elite Peers are set up, querying *Elite* becomes very simple. When a user issues a query with keyword K1, *Elite* will automatically assume by default that the user is requesting the first page of the result set associated with this keyword. Hence, a Query Message (QM) is sent in the network to the Peer holding the first page of the index, in other words PEP "K1_1", by hashing this name and setting the hash as the destination address as usual. This message contains the keyword of interest and page number as internal variables. The latter replies by sending a Query Answer Message (QAM) containing the list of ranked terms it holds and their VR & RVI values, along with the current total number of existing PEPs for this keyword. Remember that, as described in section 3.4.4.1, every PEP is aware of the total number of pages in the system for the keyword it is concerned with since the creation of a new page is automatically broadcast. The peer which issued the query thus displays to the user the VR & RVI ranked results and optionally any rippled pages it possesses if it received any, corresponding to the 3 result sets outlined in section 3.4.3. The number of additional n pages available in the result set is also displayed to the user in the event that it wishes to request another page. If this happens *Elite* simply routes the query to the correct PEP.

3.5.2 Multi-Keyword Queries

Multi-Keyword Queries follow the same pattern as single keyword queries. A QM is sent in the network using the hash of "$<K1><K2><...><K_m>$_MK" with keywords in alphabetical order. In most cases the corresponding Page Elite Peer will not exist. The Peer receiving such a query will therefore temporarily become a Multi-Keyword

PEP (PEP_{MK}) which serves the same purpose as traditional PEPs but generates its entries differently. A PEP_{MK} will stay active if it keeps receiving these QMs or die if a certain amount of time elapses without any request. They enable users to perform Boolean algebra over several indexes.

When the PEP_{MK} receives this request for the first time, it initially spawns a series of QMs for each keyword separately as described in section 3.5.1 and retrieves the index assigned to these keywords (figure 3.9). These two indexes are then merged into a single result set, according to the Boolean algebra requested, using Fagin's Algorithm[1] described in [28] (other algorithms could also be used) with a value for the variable $k = 10 * n$ (with n the page number requested). Remember (3.5.1) that by default, page number 1 is always requested. This means that the entire index for each keyword needn't be retrieved totally, only the amount necessary to rank k entries, thus limiting the number of QMs sent. In the worst case however, if each keyword Kw possesses N_w PEPs, a total of $\sum_{w=1}^{m}(N_w)$ QMs will be sent (where m is the number of keywords and N_w the number of PEPs for each keyword).

As this task involves a substantial amount of computation and messages it is necessary to minimize the initial step in creating this index as much as possible. Hence if a PEP_{MK} is kept alive after a substantial amount of time, it will subscribe to $\sum_{w=1}^{m}(N_w)$ STopics of the form $<K_w>\_n$ (with $0 < n < \sum_{w=1}^{m}(N_w)$). Whenever a ranking changes on a PEP's page, the latter broadcast this new page onto the corresponding $<K_w>\_n$ STopic enabling PEP_{MK}'s to adjust their own index and thus keep up to date. Note that since we seek to achieve maximum scalability by limiting computations and messages sent a multi-keyword query will only be sent to a single PEP_{MK} as opposed to several PEP's for single keyword queries. Thus multi-keywords aren't distributed resulting weaker security related to control over information discussed in 3.4.4. Of course, data manipulation can still be countered thanks to replication (**??**) but it is worth indicating the consequences involved. Additionally we can also expect PEP_{MK}s probability to stay alive through time to decrease as the number of keywords it contains increases.

[1]This algorithm merges 2 different lists A & B each sorted according to their score for their own particular term. Say you want to rank $k$pages from both lists; The first step involves looking at the first k elements in both list and matching those in common. If you get k entries, the algorithm stops here else, for each k entries in both lists which didn't get a match find their corresponding score in the other list and rank them this way.

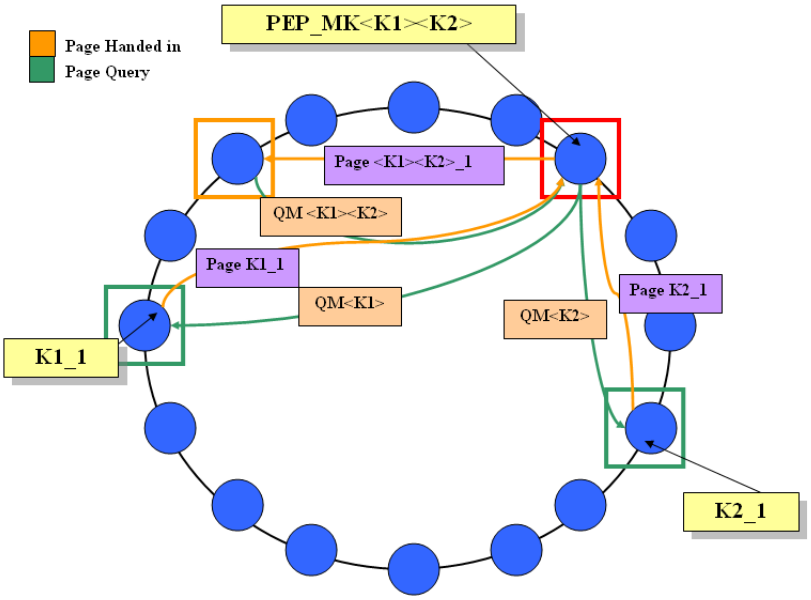

Figure 3.9: Multi Keyword Query

3.5.3 Queries Anonymity

As mentioned in section *1.3.3.3*, anonymity should be considered as one of the most fundamental requirements of a search engine. *Elite* provides query anonymity through a very simple mechanism already used and proven efficient in several systems [3]. When a query or any message is forwarded by *Pastry* nodes in the ring, any of these nodes can, if they wish at random, substitute the source address of the message with their own. Whenever a node substitutes a source address with its own, it keeps track of it and forwards it back to the original sender when it receives an answer for it. This results in the inability for any Peer to determine if the message received really did originate from this the source address Peer or not.

3.5.4 Query Caching

Also, in order to remove any bottlenecks around PEPs of any type and improve scalability, Peers forwarding QAMs keep a local copy of the page forwarded with an associated expiry time limit. Hence, if one of these Peers later on receives a request for the same page it simply answers the query without handing it to the PEP. Consequently, as a query becomes popular, the number of cached copies will increase, and as copies

accumulate, they alleviate the PEPs serving load. Additionally, if a Peer is constantly forwarding or serving the same QAMs it could also ultimately subscribe to this page's STopic as performed by PEP_{MK}s in 3.5.2

3.6 *Elite* Encryption

Finally, *Elite* uses encrypted indexing so as to "blind" any potential data manipulation in the system. In other words, if a malicious Peer did succeed in manipulating some community data, this encryption protection mechanism aims at removing this attacker from knowing what information it is actually manipulating, hence reducing potential harm induced. Allusions to this feature of the system were avoided throughout this design chapter in order to remove confusion and allow previous explanations as much clarity as possible.*Elite* provides encryption at two separate levels. In simple terms, the aim is to guarantee that PEPs of any kind, holding the indexed data, do not know what data they hold and moreover do not know themselves which PEP they are either.

3.6.1 Index Entries Encryption

Every URL or Web page mentioned in the system is always used in its cipher form, encrypted & fully decrypted by EPs providing & retrieving them to users. Hence, although PEPs store and exchange these entries, they are unable to produce any sense from them. A XOR cipher is used by EPs to encrypt these entries. Whenever referral about a URL is made to an PEP, these EPs encrypt the URL using as a key the hash of this URL($C(url) = url \oplus H(url)$).VR & RVI updates holding cipher entries can therefore openly be transferred through the Pastry network. So as to later on decrypt these entries, the EPs associate with every URL cipher a variable S consisting of the URLs hash XOR-ed with their own keyword of expertise ($S_{K_m} = H(url) \oplus K_m$).

When a Peer makes a single keyword query using K_m, it collects the corresponding entries from the correct PEP, retrieves the $H(url)$value from S using this keyword it issued and finally uses this hash value to recover the plain text url.

3.6.2 PEP Identity Encryption

Of course this first level of encryption would be completely pointless if the following mechanism wasn't in place as well. PEPs could indeed easily retrieve $H(url)$ from S by simply extracting K_m from their own names $<K_m>\_n$. Hence, whenever a PEP is mentioned, it is named instead using the hash of the keyword($<H(K_m)>\_n$). This means that PEPs can easily interact with their counterparts by changing n as they wish, but are completely unaware of which keyword their index actually belongs to since hashes are irreversible.

Furthermore, PEP_{MK}'s, named as "$<H(K1)><H(K2)><...><H(K_m)>$_MK" can still contact the PEPs needed by using their own $H(K_m)$ hashes and compare these PEPs cipher entries since they were individually encrypted irrespective of what PEPs they belonged to (using $H(url)$). A PEP_{MK} will therefore hold m S variables for each entry corresponding to the m PEPs used to form the index. Hence, a Peer initiating a multi-keyword query can use any of the m keywords issued to extract $H(URL)$ from an $S$and subsequently retrieve the plain text URL.

3.7 Ripple Effect

This section will cover in detail the ripple effect technique designed during this research which is a fundamental and intrinsic component of *Elite*. In one sentence, this algorithm aims at spreading information in the community with a speed based on its value and importance. If section 3.1 and *1.1* discussed why dissemination of information was a necessity, this part of the document explains the mechanism of how this can be done. In brief, this technique takes advantage of Elite's Peer to Peer nature to use individuals as influencers catalyzing or softening the spread of information throughout the community. Valuable information will build momentum & force, increasingly spreading through the network, while less important data will possess a ripple impact of smaller size which will be reduced or softened by the community.

3.7.1 Ripple as a Recommendation Strategy

3.7.1.1 Centrality of Awareness

In section 3.4.3, we have seen how the first 2 ranked result sets returned by *Elite* were governed by equation $V(p,t) = r * A(p,t) * Q(p)$ & $I(p,t) = (\frac{n}{r})\frac{\frac{dV(p,t)}{dt}}{V(p,t)}$. Also, as mentioned in 3.1 and 3.4.3.1 we considered the quality of a page to hold different values according to its context. However, we can assume that a page's quality for a given subject will be constant over time as long as it doesn't change. Hence, in both equations it is easy to spot how $A(p,t)$, or page p's awareness, is the only variable affecting both rankings in a specific elite group. Therefore, suppose a new page p possessing quality $Q(p) > T$ is created (where T is an arbitrary threshold), the only way possible for a system to increase both $V(p,t)$ & $I(p,t)$ is by artificially increasing the speed at which $A(p,t)$ increments. Within in our own environment, if an Elite Peer already aware of p ergo $Q(p)$, judges that this page's VR & RVI values do not come near their full potential, it can only be because $A(p,t)$ is closer to 0 than 1, hence it should recommend this page to its peers.

3.7.1.2 Zero input Recommendation

Recommendation solutions aren't a new idea. Several attempts to use this type of solution to increase awareness of new pages have already been implemented [5]. However, nearly all of these are based on models where users must explicitly input recommendations themselves. These systems are thus limited to how much effort individuals are willing to make and hence do not scale. Elite however, is built as a zero input recommendation system where reference to pages are initiated automatically by analyzing user behavior.

As we have shown in section 3.4.3.1, since direct measurement of page quality is impossible, the local visit rate of a page is very close in determining how a single peer values a web page ($V_L(p,t) = r * Q(p)$). Furthermore, assuming a page's quality is known based on the previous statement and that similar peers will value it equally, the elite group (or global) Visit Rate variable ($V(p,t) = r * A(p,t) * Q(p)$) can then be used by a single peer in determining the relative community awareness of a page by

inserting $V_L(p,t)$ in the latter equation giving:

$$A(p,t) = \frac{V(p,t)}{V_L(p,t)} \tag{3.3}$$

.

On a regular basis, Elite Peers thus scan through their web page set related to their keyword of expertise and use their local visit rate value $V_L(p,t)$ in conjunction with an approximation of the pages global value $V(p,t)$ to compute $A(p,t)$ using equation 3.3. An EP can easily estimate $V(p,t)$ based on the knowledge of page's PEP mappings 3.4.4.2 and their lowest entry values 3.4.4.4. If the estimated elite awareness value is below an arbitrary threshold and the quality assigned locally by this peer is above a certain criteria, this local Peer can then initiate a ripple recommendation without having to ask the user.

3.7.1.3 *Elite* Ripple Spread & Ripple Size

When an Elite Peer decides to ripple a page (it can only ripple a specific page once), it anycasts a Ripple Web page Message (RWM) containing this page's entry, to other EPs on the $<K_m>$ STopic (where K_m is its keyword of expertise). The message will hop through any EP *randomly* (figure 3.10). When an EP receives a RWM containing a reference to a page it has already been recommended, it increments the number of recommendations it received for it and forwards the message to another EP. Incrementing the number of recommendations received for a page increases it's rank in the ripple result set entries (3rd column 3.4.3.3) thus increasing the probability of it been viewed by users according to [36, 47]. If this is a page which hadn't been recommended to it already, it acknowledges the recommendation. The algorithm stops when the page's recommendation has been acknowledged s amount of times or when the message has gone through all EPs. s represents what is called the ripple size. It is the number of EPs which had not previously received any recommendation for this page and are currently unaware of it.

By setting s to a relatively low value, for the whole Elite community to be recommended this page, a substantial amount of peers must ripple this page. This enables pages to be spread around the community with a speed proportional to their quality estimated as a group as opposed to only one peer. Peers as a group can then

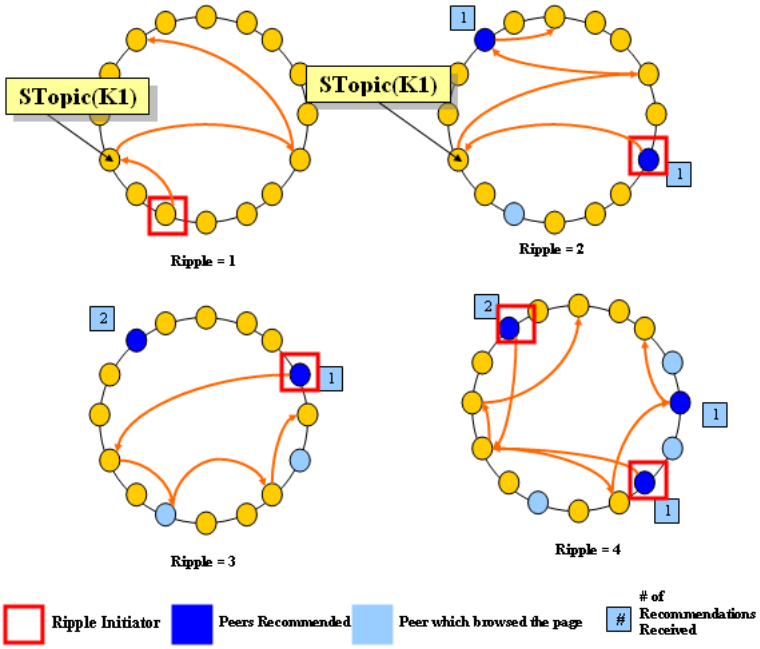

Figure 3.10: Ripple of Page p with $s = 2$

determine which information is worth to spread and which isn't. Remember that a ripple is initiated if a peer estimates that a page possesses a certain amount of value but isn't known as much as it should be by the community. This is a personal choice made locally by a Peer. If this view isn't shared by most of other EPs, these won't initiate any ripples for this page hence not incrementing its recommendation value on other Peers. If they do ripple this page however, the latter will be ranked high in EPs rippled result sets currently unaware of it increasing the chances of it been aware. This creates a snow ball effect for high quality pages increasing their possibility of getting awareness and in the contrary damps the spread of other pages with lower quality. The community hence as a group controls how information is spread in the network.

3.7.2 Ripple as a Censorship Model

Section 3.7.1 described how the Ripple Effect could be used as a recommendation system to articially increase a page's awareness. This section however, describes how *Elite* attempts to achieve censorship throughout the community through democratic

decisions.

Censorship through reporting is becoming very wide spread nowadays over the net with organisations such as Wikipedia or Picassa enabling their users to report bad content back to service providers. Although, this technique must function to some extent since an increasing number of Web content providers are adopting it, it still only relies on the good will of web users to make the "effort" of reporting bad content from a purely altruistic mind set. Non zero input systems, as we saw previously (section 3.7.1.2), only scale to a certain extent, hence the efficiency of this method is doubtful. Although, in this situation a zero input strategy is impossible since determining content suitability can only be done by people, a better approach would consist of rewarding users reporting bad content.

Moreover, at the end of the day, whatever reporting is done by web users, the organisation is still the sole entity responsible and holding the power to decide what should or should not be censored. As has been pointed out in 1.3.3.4, this raises many ethical issues which should be avoided as much as possible.

3.7.2.1 "Report & Support" and "Tit for Tat" Censorship

Elite's censorship approach is based on a "Report & Support" combined with a "Tit for Tat" model. The overall idea behind this is the following.

Whenever a web user browses over content which it feels goes against universal values, (such as child pornography, racism advocacy etc...) it alerts the community by reporting it. Doing so spreads a censorship request to other members of the community who either support or neglect it. If, in a given amount of time, a censorship request receives support from the community above an arbitrary threshold, the author of this information is notified that its page is being blacklisted from the community's information commons and removed from the ranked list. Note that the page isn't actually removed from the Web, since *Elite* as we pointed out for search engines in general (section 1.2), should only *present* web information to users and not control it. However, as the page is removed from the list presented by the search engine, its accessibility is reduced to a minimum. Moreover, if it was somehow decided, that blacklisted pages should be completely removed from the Web, we could very well imagine the application notifying a central legal authority to do so.

Secondly, so as to reward users for their effort in deciding how to manage such information, a "Tit for Tat" system could make sure reporting alerts attract as much attention to users as possible. In other words, if a user contributed in supporting, or voting, to remove Web content going against universal values and that the community decided so as a whole, we can assume that the information was indeed bad content, therefore this user should be rewarded for having contributed to this task. A sort of point system could hence be induced, where the more a user helps the community tackle content going against universal values, the more the user should possess the ability to alert the community. So for example, if this user somehow, discovers that private information about itself which it didn't want to be made public is somehow released on the net, it could initiate such an alert to protect it's privacy rights and ask the community to support it. Hence, the more you help others protect their privacy rights, the more you possess the ability to protect your own.

Finally, to avoid malicious peers repeatedly initiating alerts for no reason in particular and hence overflowing users with dummy requests, if an alert doesn't achieve enough support in a given threshold time, we could assume this alert to be a dummy one and hence remove "points" from this user to avoid it repeatedly doing so.

3.7.2.2 *Elite*'s Censorship Functionality

Since *Elite* is built on Peer 2 Peer technology, it's ability to enable people to make decisions together is made very easy. Thanks to a combination of both *Pastry* and the *Ripple Effect*, democratic decisions within the community are possible without the need for any external authorities.

Whenever a peer initiates an alert, as described in the previous section, a Censorship Alert Message (CAM) is sent through *Pastry* (with as a destination id the hash of <*url*>_CAM, where url is the *url* of the page wishing to be censored), along with the initiation of a ripple with a Support Censorhip Alert Message (SCAM) containing the specified *url* (as shown in figure 3.11).

The node receiving such a message becomes temporarly the "chair" of this censorship commity. It doesn't hold any power to make any decisions concerning this alert but can only organise the decision making. Peers receiving SCAM ripples however are the only entities capable of making decisions (the original alerter can only

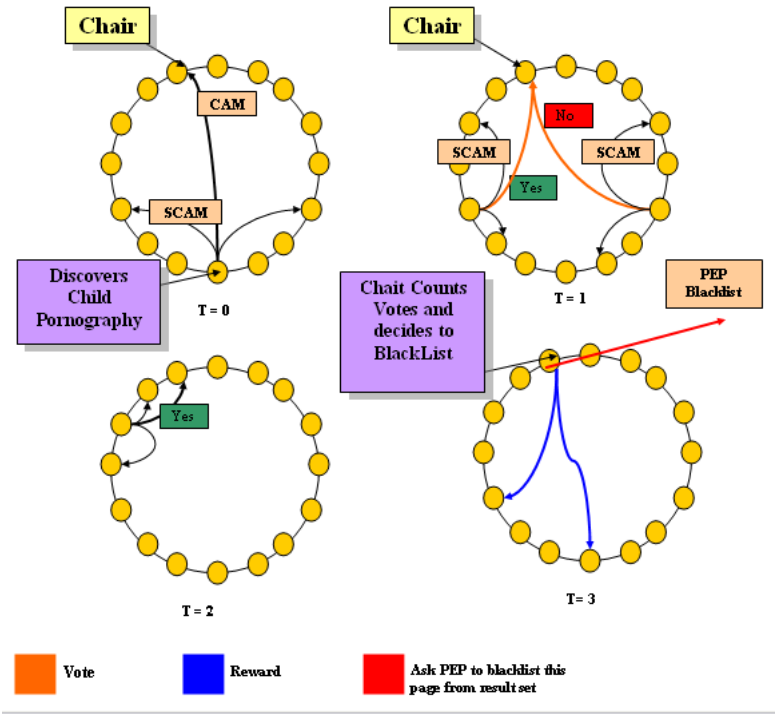

Figure 3.11: Censorship and Ripple

request a ban it doesn't hold the power to make this decision). Whenever such a message is received, users can either agree, disagree or neglect it. If the latter occurs nothing happens, whereas if a decision is made, a Censorship Decision Message (CDM) is sent via *Pastry* to the chair with <*url*>_CAM as a destination Id and the ripple is spread to more random members so as to attract more attention. Thanks to the ripple effect's nature 3.7, the more ripples received for a particular issue, the more this issue will be ranked higher when presented to users and hence increasing the likelihood of it being seen [36, 47]. The chair receiving a CDM, temporarly keeps the source Id of this message to enable potential feedback later on and counts the votes. Whenever it receives a number of votes above a certain threshold, it decides to either blacklist the page, or not, according to the outcome of the vote by contacting the PEP holding this page using the same method used by EPs or PEPs 3.4.4.2. It also hands in positive feedback (possibly as points) to users who contributed to making this decision. If the decision was positive, the initial peer who alerted the community is given positive feedback whereas if it was negative, (depending on how overwhelmingly), the peer receives negative feedback. Peers who made the decision contrary to the majority do not receive any negative feedback as they possess their own right to hold this view. Once this whole process is finished, the peer acting as the chair, removes itself from this position and remains a "Simple Peer" again.

3.7.2.3 Minority Opinions in a Majority Dominated World

Even though *Elite* enables democratic decisions to be made by the community, an important fact shouldn't be neglected. As Winston Churchill puts it so famously well, "*It has been said that democracy is the worst form of government except all the others that have been tried*". In other words, although democracy has been a positive step foward in our society, we need to keep in mind that it isn't perfect. This subject is out of the scope of this document of course, but one important issue however worth noting, is how the majority tends to dominate entirely the rest of the population. This should be balanced somehow by protecting views and opinions shared by a substantial minority of the people, hence promoting diversity of content as encouraged in our code of ethics (section 1.1). As an example, important concepts which we hold today as fundamental (such as the abolition of slavery, death penalty...) weren't always

shared by the majority. Hence, a mechanism protecting challenging views shared by a substantial minority of the population (how much substantial is open to discussion) should be in place in order to counter any possible "oppression" by the majority.

This could easily be achieved within *Elite*, by rippling a sort of "Idea Protection Request" message to the community which would seek to find this substantial minority for opinions which authors would be aware aren't yet shared by the majority. If such a large minority was found it would be very easy then, to simply assign to any peer the responsibility of listing these protected pages so that an individual initiating a CAM message would be notified that this information or opinion is shared among a large minority of the global population and hence shouldn't be censored.

3.8 Future Design Consideration

Although, this chapter aimed at providing solutions to issues raised in Chapter 1, improvements to this design are encouraged in future research or deployment.

As a simple example, so as to maximise scalability for instance, stopwords should be incorporated in the system to minimise network costs. Stopwords[26] consist of nouns such as "a", "it", ... which are so commonly used in the english language that they do not carry enough weight by themselves to provide meaningful IR information. Enabling such keywords to create elite groups of their own and all associated entities would be extremely waistful. This issue could be very easily be solved within Elite by assigning to specific peers the responsability of holding a list of stop words for each language in the system. Previous any KEP creation, a consultation to these peers would be necessary to ensure the keyword specified isn't a stopword.

Moreover, and a lot more challenging, would be to open the concept of "Eliteness" to the entire population of peers. *Elite*'s original design aimed at not distinguishing any Elite group within a given keyword but instead incorporating potentially the whole community in a semantic structure in the form of a spiral (or a never ending open ring) as opposed to a closed ring. Peer's expertise wouldn't determine their belonging to an Elite group but instead would specify their position in this spiral. As a peer's expertise in a given keyword would increased, it's position in the spiral would become closer to the centre. This also would suggest that, so as to regulate the network flow of VR updates in the system towards PEPs, each peer's updates importance or frequency

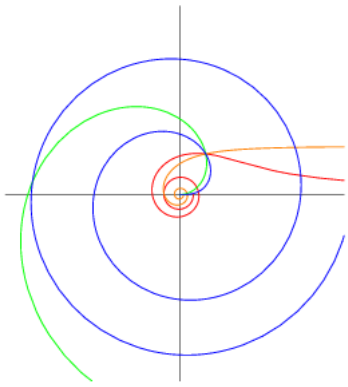

Figure 3.12: *Elite* Archimedean Spiral Semantic Architecture

would be assigned a weight proportional to it's direct distance towards the origin. If we considered an archimedean spiral for example, with equation $r = \alpha\theta^{\frac{1}{n}}$, where r is the radial distance, θ the polar angle and n a constant, the inverse of variable r would be used in specifying both the expertise of a peer, the weight of its VR updates as well as their frequency. *Elite*'s semantic architecture would result hence in a structure very similar to figure 3.12where every peer could be located, if it wished, in different spirals corresponding to different keywords, in a different position related to its own expertise in these keywords.

Chapter 4

Elite Modelling & Prototyping

The following chapter presents to the reader our experiments and protype implementation. Section 4.1 outlines the derivation of the *Ripple Effect*'s formal model, while section 4.2 describes how a simulation of this technique was carried out. Section 4.3 finally, presents in detail *Elite*'s prototype implementation.

4.1 *Ripple Effect* Formal Model

Preceding any prototyping, a correct understanding of the underlying laws governing the *Ripple Effect* was desirable. The first step therefore consisted of creating a formal model of this algorithm so as to study its behaviour and properties. The formal study sought the likelihood of an Elite Peer, previously rippled by page p, of being repeatedly rippled with the same page by different Peers. This behaviour is crucial as it prescribes the probability of a rippled page incrementing the number of recommendations it received and hence increasing its local rank on an Elite Peer. Equation

$$HRip^{s,n,u}(R) = \sum_{T=s}^{Rs} \left\{ P1^{n}(T) * P_{s=1}^{n,u}(T) + \sum_{i=1}^{T-1} \left[P1^{n-i}(T) * P_{s-1}^{n-i,u-s-1}(T-i) * Spw_{s=1}^{n,u}(T) \right] \right\} \tag{4.1}$$

was found to describe this behaviour, where n is the number of Peers in the Elite group, u the number of Peers currently unaware of page p, s the ripple size and R the number of times this ripple is being initiated $\forall R, T > 0; s, n > 1$.

The rest of this section describes in details how this formula was elaborated.

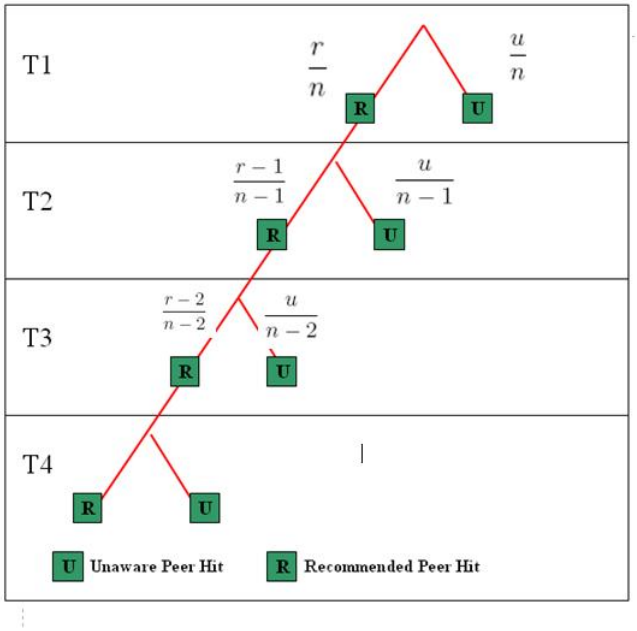

Figure 4.1: Ripple with $s = 1$

Readers interested in the formal evaluation of this behaviour can skip this section and directly read section *5.1* in chapter *5.*

4.1.1 Ripple, a Series of Trials

Rippling a page consists of a specific set of trials. Recall from section **??** how an EP receiving a ripple message can either decide to acknowledge it, if it hadn't been recommended this page before, or forward it, if already had this page in it's ripple set. Furthermore, it is worth remembering how the ripple algorithm stops when the number of peers acknowledging a ripple is equal to its ripple size s. In an elite group consisting of n members with r already possessing this page p in their rippled set and u members currently unaware of it, figure 4.1 shows how this algorithm can be modeled as a binary tree where $n = r + u$.

4.1.2 Formal model of Ripples of Size 1

Initially, for clarity of explanations we will only consider a ripple of size 1. The following part of this section (4.1.3) will then generalise the model for any size s.

4.1.2.1 Hitting Previously Recommended Peers

Assuming a ripple is launched, from the diagram, it is easy to notice how, in a group containing $n = r + u$ peers, the probability for a ripple hitting a peer r already been recommended this page is $\frac{r}{n}$ while the opposite if $\frac{u}{n}$. Since, a peer already visited in a previous trial cannot be revisited twice, at each new trial the number of r peers is decremented by 1. Hence assuming a second trial is launched, the probability this time of a ripple hitting an r peer becomes $\frac{r-1}{n-1}$ and so on. Remember that since we set the ripple size $s = 1$, whenever a ripple encounters a u peer, the algorithm stops. Hence while the number of r peer decreases at each trial, the number of u peers a ripple could meet however stays constant over the entire set of trials. Therefore in the second trial the probability of hitting a u peer is $\frac{u}{n-1}$. The probability $R(T)$of a ripple hitting a previously recommended peer at *any trial* T can thus be generalised to:

$$R^{n,u}(T) = \frac{r - T + 1}{n - T + 1} \tag{4.2}$$

with $T \geq 1; r, n \geq 0; r = n - u$.

4.1.2.2 Trials Probability

Being aware of the fact that with a ripple size $s = 1$ the algorithm will stop as soon as it meets a u peer, the question of the probability of a ripple to launch T trials naturally emerges. From the diagram it becomes evident that, in order for trial T to occur, all of the previous trials outcomes must have hit a r peer. Hence, the probability of having trial T is equal to the probability of hitting a r Peer in all trials $T-1, T-2, T-3...$ The probability $P(T)$ of a trial T occuring is a recursive function of the form:

$$P_s^{n,u}(T) = R^{n,u}(T) * P_s^{n,u}(T-1) \tag{4.3}$$

which can also be written as:

$$P_s^{n,u}(T) = \prod_{i=0}^{T-2} \frac{r - i}{n - i} \tag{4.4}$$

where $P(T) = 1 \forall T < s \& r = n - u$

4.1.2.3 Probability of one particular Peer being Hit

Moreover, it will be interesting to know the probability of a ripple hitting one particular Peer at *any Trial*. This is very straigth forward to understand. Say trial 1 begins. The probability of a ripple hitting one particular peer among a group of n Peers is $\frac{1}{n}$. In the second trial, as stated in 4.1.2.1, one peer is removed from the probability of being hit again hence the new probability of one particular peer being hit in this trial becomes $\frac{1}{n-1}$ and so on... The probability $P1(T)$ of 1 particular peer being hit at *any trial* is therefore summarised as:

$$P1^n(T) = \frac{1}{n - T + 1} \tag{4.5}$$

4.1.2.4 Probability of one particular Peer being Hit at a Trial

Consequently, we now want to know what is the probability $H_s^{n,u}(T)$of one particular Peer being hit at the T^{th} trial. This is the same thing as asking ourselves, what is the probability of one particular peer being hit at trial T and what is the probability of this trial occuring? Using equation 4.4 & 4.5 we get:

$$H_s^{n,u}(T) = P1^n(T) * P_s^{n,u}(T) \tag{4.6}$$

which can also be written as:

$$H_s^{n,u}(T) = \frac{1}{n - T + 1} * \prod_{i=0}^{T-2} \frac{r - i}{n - i} \tag{4.7}$$

where $\forall s, T \geq 1$

4.1.2.5 Probability of getting Hit in 1 Ripple

Once we know the likelihood of a particular Elite Peer being hit in any trial, it will be also useful to know the probability of an EP being hit in a single ripple. This is equal to the summation of the probabilities of this peer being hit in any trials that occured within a ripple, giving us:

$$HRip^{s,n,u} = \sum_{Min(T)}^{Max(T)} H_s^{n,u}(T) \tag{4.8}$$

For $u > s$, the minimum amount of trials T a ripple could initiate is equal to the number of peers it must hit, unaware of page p which corresponds to the ripple size:

$$Min(T) = s \tag{4.9}$$

Additionaly, a ripple will launch a maximum amount of trials if it happens to hit all r peers which were previously aware of p and then hit $s$unaware peers consisting of:

$$Max(T) = r + s \tag{4.10}$$

Equation 4.8 hence becomes:

$$HRip^{s,n,u} = \sum_{T=s}^{r+s}(\frac{1}{n-T+1} * \prod_{i=0}^{T-2}\frac{r-i}{n-i}) \tag{4.11}$$

4.1.2.6 Probability of getting Hit in any Ripple

Finally, since we know the number of Elite Peer memberships in the system, and assuming we knew the number of times a page had been rippled in the system, we would know the number r of Elite Peers who where introduced to this rippled page. The latter is very easy to comprehend since each ripple will hop along EPs until s (s is the ripple size) of them are newly introduced to this page p. If ripple R is being initiated, we know that $r = s * (R-1)$ Elite Peers have been introduced so far to this page by rippling. This relation, of course, also assumes that page p is only discovered by other Elite Peers through rippling which isn't necessarily the case in real life (they could simply browse randomly on it) but the probability of this event occuring is so low that it is considered negligible for the time being. Knowing the latter, it is possible to model the system in relation to the amount of times R a page p is rippled. Using the latter equation as well as equation 4.11, we get:

$$HRip^{s,n,u}(R) = \sum_{T=s}^{Rs}(\frac{1}{n-T+1} * \prod_{i=0}^{T-2}\frac{r-i}{n-i}) \tag{4.12}$$

$\forall R, T \geq 1$

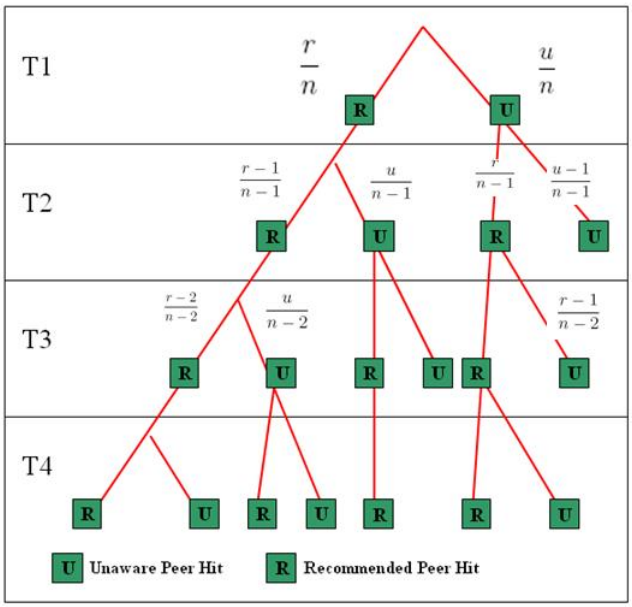

Figure 4.2: Ripple with s = 2

4.1.3 Ripples of any Size

Lets consider now the case when the ripple size $s > 1$. If $s = 2$ for example, in order for the algorithm to stop, the ripple must hit 2 Elite Peers which where previously unaware of this page. Hence the binary tree depicted in figure 4.1 is now extended to produce a new tree illustrated in figure 4.2. As you can see, whenever the ripple hits a peer of type u for the first time, the algorithm continues in a similar way as a ripple of size 1. This child ripple however, possesses a different set of variables n, r and u corresponding to those in the system at the moment when it was produced, minus a u peer (since the original reason for this tree to be created is because a u peer was encountered). This graph can also be seen as a "Parent" ripple with size $s = 1$ spawning "Child" ripples of size $s = 1$ themselves.

4.1.3.1 Trials Probability

The likelihood of having T trials now becomes the probability that the ripple hasn't hit any u peer previously (as in 4.1.2.2) or the probability that it hit only 1. The second statement could also be formulated in the following way. The probability that

a ripple only hit 1 u peer and no more, is equal to the probability that a ripple of size s=1, with the corresponding variables stated in section 4.1.3, was initiated and that within this nested ripple, no u peer was hit. Moreover, recall that at each trial, this "Parent" ripple of $s = 1$ initiates a new ripple of $s = 1$, hence the probability $Spw_s^{n,u}(T)$ that it initiated a new ripple of size $s = 1$ at trial T is equal to the probability that the "Parent" ripple hit a u peer at this trial and hadn't done so previously, giving the following equation:

$$Spw_{s=1}^{n,u}(T) =\sim R(T) * P_{s=1}^{n,u}(T) \tag{4.13}$$

where $\sim R(T)$ is the inverse of $R(T)$

Now we want to know the probability of this "Child" ripple of size $s = 1$ of having previously hit only r peers at trial T. If we consider only this "Child" ripple spawned at trial i, this probability is equivalent to $P_{s=1}^{n-i,u-1}(T-i)$. Hence the probability that a ripple with $s = 1$ hasn't hit any u peer previously or only 1 in any of the "Child" ripples spawned at each trial for all s is:

$$P_s^{n,u}(T) = P_{s=1}^{n,u}(T) + \sum_{i=1}^{T-1} P_{s-1}^{n-i,u-s-1}(T-i) * Spw_{s=1}^{n,u}(T) \tag{4.14}$$

where $P_0^{n,u}(T) = 0 \& P_{s>0}^{n,u}(1) = 1$

4.1.3.2 Probability of a particular Peer being Hit at a Trial

Now applying the same formula (equation 4.6) as in section 5.1.3 in order to get the probability of one particular Peer being hit at a trial$T$, gives:

$$H_s^{n,u}(T) = P1^n(T) * P_s^{n,u}(T) \tag{4.15}$$

$$H_s^{n,u}(T) = P1^n(T) * P_{s=1}^{n,u}(T) + \sum_{i=1}^{T-1} \left[P1^{n-i}(T) * P_{s-1}^{n-i,u-s-1}(T-i) * Spw_{s=1}^{n,u}(T) \right] \tag{4.16}$$

4.1.3.3 Probability of getting Hit in any Ripple for any s

And finally, using equation 4.12 we get:

$$HRip^{s,n,u}(R) = \sum_{T=s}^{Rs} \left\{ P1^{n}(T) * P_{s=1}^{n,u}(T) + \sum_{i=1}^{T-1} \left[P1^{n-i}(T) * P_{s-1}^{n-i,u-s-1}(T-i) * Spw_{s=1}^{n,u}(T) \right] \right\} \tag{4.17}$$

$\forall R, T > 0; s, n > 1$

4.2 *Ripple Effect* Simulation

4.2.1 Simulation Objective

Having analytically formalised the *Ripple Effect* algorithm, the subsequent question naturally arises. How are pages' awareness evolution over time affected in an environment of this kind? A computer simulation of such a scenario was therefore implemented, modelling user browsing behaviours over a set of web pages constantly changing in time. These series of experiments sought to understand how *rippling* would affect the popularity evolution of pages as well as the general overall quality of user web browsing. The simulations were therefore carried out in both "ripple" and "non ripple" environments in order to compare resulting figures. Additionally, an understanding of the impact of ripple size as well as minimum quality criteria thresholds related to rippling pages (section *3.7.1.2*) was highly desirable so as to choose optimal parameters in an *Elite* prototype. We expect the rippling of newly created pages to improve the general web browsing quality as well as increasing the speed at which these pages gain popularity thus reducing entrenchment effects.

4.2.2 Simulation Metrics

In order to evaluate these aspects mentioned previously, the awareness evolution of every page in the system was recorded. From these figures, a Time to Become Popular (TBP) metric was derived. The TBP value measures the time taken for a page to reach 90% of popularity according to Cho's definition of popularity P(p,t) (section 2.1). A minimal TBP value for each page as well as a maximum amount of pages reaching this value is sought.

Secondly, in order to measure the overall quality of web browsing, a Quality Per

Click metric outlined in [56] was used, measuring the average quality of pages viewed by users, amortized over a long period of time. It is computed using the following formula:

$$QPC = lim_{t\to\infty} \frac{\sum_{t_1=0}^{t} \left\{ \sum_{p\in P} (V(p,t) * Q(p)) \right\}}{\sum_{t_1=0}^{t} \left\{ \sum_{p\in P} V(p,t) \right\}} \tag{4.18}$$

where P is the set of web pages in the system. We seek to have a QCP value as high as possible. Moreover, the number of ripples initiated as well as messages sent per ripple was measured. The latter variable gives us the amount of peers incrementing their recommendations stored in their ripple web pages set. Furthermore, all of these variables will be measured as a function of ripple size s and minimum ripple quality $RipMin$ respectively modulated between values 1 to 5 & 0.5 to 1. RipMin is only considered for quality values equal or above 0.5 since we are only interested in promoting pages of better quality than average. Finally, the special case of a variable ripple size s proportional to the quality of the page being rippled will be taken into account as well.

4.2.3 Default Scenario

It is probably worth emphasizing how the environment we seek to model is highly complex and dynamic. This system is composed of search engines, users interacting and making decisions, an immense set of pages evolving through time which we need to trace and so many other random variables which would be impossible to attempt any control of. Even if we owned the most popular search engine, clean room experiments would be impossible since previous experiments would affect the subsequent ones and so on...The only way such a system can be studied is by analytical models and simulations which can only achieve tradeoffs between precision of measurements and implementation feasibility within a given accessible framework. Hence the following paragraph attempts to formulate assumptions which aren't perfect but try to preserve as much as possible the essence of the system studied. Each assumption will therefore be based, as much as possible, on real life web analysis of previous researches.

Recall in our model how peers are interested in different subjects, and pages possess different qualities for a specific set of these subjects. Moreover, we mentioned previously as well how Elite Peers of a particular group are all interested in a given

Symbol	Meaning
P	Set of web pages
U	Number of EPs within an elite group
ζ	Number of pages visited per day by each EP
ψ	Probability of randomly surfing the web
γ	Teleportation Probability
$PH(x)$	Probability of a link with rank x of being clicked based in a result set
$RipMin$	Minimum quality value for a page to be rippled
s	Ripple Size
K	Set of Web Pages Peers are initially aware of
l	Life time expectancy of a Web page
λ	Web page renew Poisson process rate parameter

Table 4.1: Notation Summary used in this simulation model

subject and that ripples only occur within these groups. Hence, our model will consider a set of P web pages within a set of users of one elite group of size U interested in one particular subject. We consider throughout this section the quality of a page as its quality in respect to the given subject of interest (To ensure as much clarity as possible, a summary of notations is provided in table 4.1).

In our model, pages are assigned a quality $Q(p)$ uniformly distributed from 0 to 1. The set of pages isn't fixed through time but evolves due to new pages being created and others deleted. In order to keep this experiment manageable, it was assumed that the amount of new pages matched those being deleted keeping a constant set of P pages over time set at about 10 times the number of Peers. Retirement of pages was modeled according to a Poisson process with rate parameter λ, and hence the expected life time of a page is $l = \frac{1}{\lambda}$ a value which was set as $l = 1.5$ based on data from [8]. New pages created were assigned the same quality as those dying, guaranteeing a constant quality for our web page set. When a new page is created it's initial awareness and popularity is equal to 0.

According to report [59], each Elite Peer in the system visits on average $\zeta = 51$ pages per day. A peer is known to browse the web in the two following ways: it either randomly surfs[1] the Internet with a probability ψ or else visits pages which were returned by a search engine with probability $1 - \psi$. According to [59], ψ is on

[1]Section **??** gives more information on Random-surfing modelling

average assigned a value of 0.63. However, as this experiment seeks to compare how an environment using the *Ripple Effect* performs with respect to one consisting of a pure popular based search engine, so as to maximize the quality of our comparisons, it is in our interest to assign ψ the value for which a "non ripple" environment would perform the best. Ergo, a small set of experiments will be needed in other to find the correct value for this variable which will be described in the following chapter. Additionally, while performing random surfing, web users traverse a neighbor link with probability $1-\gamma$ and jump to a random page with probability $\gamma = 0.15$. This constant γ is known as the teleportation probability [50].

The system records global page visits and a search engine ranks these pages regularly according to their visit rate. In this model our search engine was modeled using 2 columns consisting of the VR (visit rate or most popular column) and ripple result set. We assumed the likelihood of a peer choosing one of the two columns as been equal and modeled the probability of a link, with a specific rank in either of these 2 lists, being click using the following formula derived in *[36, 47]:*

$$PH(x) = \frac{\zeta * (1-\psi)}{\sum_{i=1}^{P} i^{\frac{-3}{2}}} * x^{\frac{-3}{2}} \tag{4.19}$$

where x is the rank position of the page.

Time is divided into discrete units and at the end of each one, peers ripple pages (with a quality equal or above $RipMin$) if they do not currently achieve their estimated global full potential according to each individual peer's point of view (section 3.7).

Finally, so as to guarantee the reliability of our results, measurements were taken when the system reached a steady state behavior. As the system needs to be launched with an initial amount of web pages P, we needed to ensure that the awareness evolution of pages measured were within an environment fully affected by rippling so the steady state behavior was considered as been reached when the initial set of web pages (unaffected by rippling for some part of their life time) had all died.

4.2.4 Simulation Program Architecture

This part of our simulation description will go through the main components of the implementation. The first paragraph will give a brief description of each module and

their function while the second will outline the algorithm used during simulation.

4.2.4.1 Architecture Overview

As figure 4.3 illustrates, our simulation implementation consists of 5 major components consisting of a WWW page corpus, an Elite surrogate, a set of Peers, a Global Data Collection module and finally, a SQL database storing all the data needed for future evaluation.

The first most important module consists of the simulated WWW pages corpus holding a set of P Web pages objects. Web pages were modeled using inheritance with two classes *Web Page Ref* & *WWW Web Page*. The first parent class simply represents a reference to any web page (which is used between peers). This reference holds the URL of this page and its associated page quality for the given subject of interest to the Elite group. The child class however, models the web page from a global perspective. It tracks its global awareness evolution through time as well as its age and TBP value to be reached. The second class is only needed to represent our World Wide Web surrogate while the first is used whenever a reference to one of these pages is needed.

The second important components in the system are the Peer objects, each representing and modeling a single Peer's behavior and Web knowledge. These are the main agents browsing the Web and hence affecting the awareness distribution, and search engine ranking in the system. They hold 2 sets of page references consisting of those they are aware of already and those which were recommended to them by ripple but which they aren't aware of yet. These two sets simply consist of *Web Page Ref* objects from original pages contained in the global corpus.

The Elite surrogate is the core of our implementation holding each component together and acting as the central figure in the system. It manages the birth and death of Web pages, as well as simulating the ripple message passing in the Elite group since it holds a reference to each Peer modeled. It also records Web pages visits from peers, ranks these pages according to their visit rate and hence returns when needed the correct page ranking to peers requesting it.

Finally, the last 2 components serve a purely technical goal consisting of gathering all the data needed later on for analysis. The Global Data Collection module measures the number of ripples launched as well as the number of messages sent and

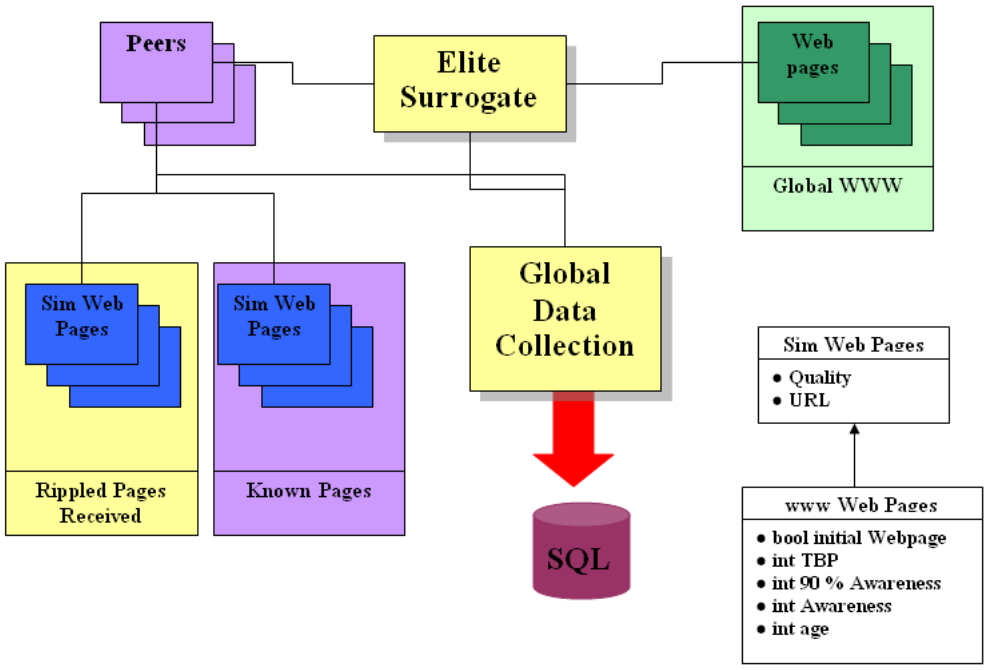

Figure 4.3: Simulation Architecture

total amount of visits that occurred. From all of this data it is therefore capable of estimating a QCP over the total length of the experiment. All of this information is then loaded into the SQL database which acts as a dead end data funnel.

4.2.4.2 Algorithm Description

When the simulation is launched for specific $s$and $RipMin$ variables, an initial global web page corpus is created containing P pages uniformly assigned specific qualities (from 0 to 1) and age. K of these pages are initially randomly assigned to peers and represent pages they are initially aware off when the experiments starts.

For each time unit, the Elite surrogate calls the $visitPage()$method on all Peers which initiates user browsing behavior. Peers distributeζvisits in a time frame corresponding to a day as follows:

$1 - \psi$ of these visits are performed through *Elite* search queries. So as to retrieve the correct current result set in the system, a Peer calls the getPopularRankedURLs() method on the *Elite* surrogate which returns the

most popular ranked pages and associates these with the set of rippled Web pages it received through previous rippled message (ranked in increasing recommendation order). Once it holds these ranked pages it chooses either of these ranking with equal probability and uses equation 4.19 to choose one of them.

$\psi * (1 - \gamma)$ of these visits are made through random browsing of previously known pages. These visits are uniformly distributed through a Peer's set of known pages in proportion to their quality based on equation $V_L(p,t) = r * Q(p)$ in section *3.4.3.1.* Hence a page of quality $Q(p) = 0.6$ will be in general locally browsed twice as much as a page of quality $Q(p) = 0.3$.

$\psi * \gamma$ of these visits are made totally randomly by calling the getRandomURL() method in the Elite surrogate which returns one page totally at random from the global corpus, thus modeling the teleportation behavior encountered in real life.

Each time a visit is made, Peers inform the *Elite* surrogate and the Global Data Collection module about it. It specifies if this was the first time they visited this page or not enabling Web pages in the global corpus to keep track of their global awareness evolution. The *Elite* surrogate then adjusts its global visit rate ranking according to visit information it receives.

At the end of each time unit, peers estimate which pages should locally be rippled (based on their quality and global rank position from section 3.7) and hand these pages over to the *Elite* surrogate which simulates the ripple message passing between Elite Peers. The latter also replaces any pages having passed their lifetime expectancy l by a new set of pages with mapped qualities to ensure a consistent global quality of the corpus. When a page dies, its awareness evolution over time as well as it's TBP value (if it reached it) is loaded in the database.

Finally, once the experiment has ended, the Global Data Collection module inputs in the database, the total number of ripples initiated along with the number of messages and calculates the estimated QCP for this experiment based on visit data received through the experiment by Peers.

4.3 *Elite* Prototype Implementation

Following these experiments, an initial functional *Elite* prototype implemented in *Java* was completed and tested. This section will cover in details it's functional design and architecture. All the features mentioned in chapter 3 were implemented with the expection of encryption and censorship (due to time restrictions) as they are not essential for this proof of concept. During the course of this research, *Elite* has evolved and is now the central figure of an open-source community project freely accessible to users on the Web. The *Elite Project[43]* home page can be found at :

http://sourceforge.net/projects/elite/

4.3.1 *Elite* as an Internet Layer

Elite is built as a second layer application within a 3 tier framework. It was designed to fit underneath any popular Internet browser and is omnipresent as well as transparent to users once it is installed. It is supported by an underlying layer consisting of the *FreePastry* substrate. *FreePastry*[68] is an open-source implementation of *Pastry*[52] developed by Rice University [1] in Texas and originally intended to be used as a tool allowing researchers to evaluate and perform development in Peer to Peer substrates. It has subsequently matured and has now become suitable for full scale Internet deployment.

4.3.2 *Elite* Architecture Overview

Elite's overall architecture is very straight forward (figure 4.4). It comprises of 3 main modules consisting of a *Web Page Corpus*, a variable number of *Front Ends* and a *Page Cache*. As mentioned in the design chapter, the corpus only retains a list of Web page objects with their associated Vector Space Model holding their main subject of interest & strength's. Actual Web pages needn't be stored at all, removing the need for large storage space. The sole purpose of the Page Cache however, is to store result set pages retrieved or forwarded in the network. It aims at reducing the load on highly popular queries as stated in section *3.5.4*. It's size is variable and scales itself according to the node's storage capacity. The Front End module on the other hand, probably

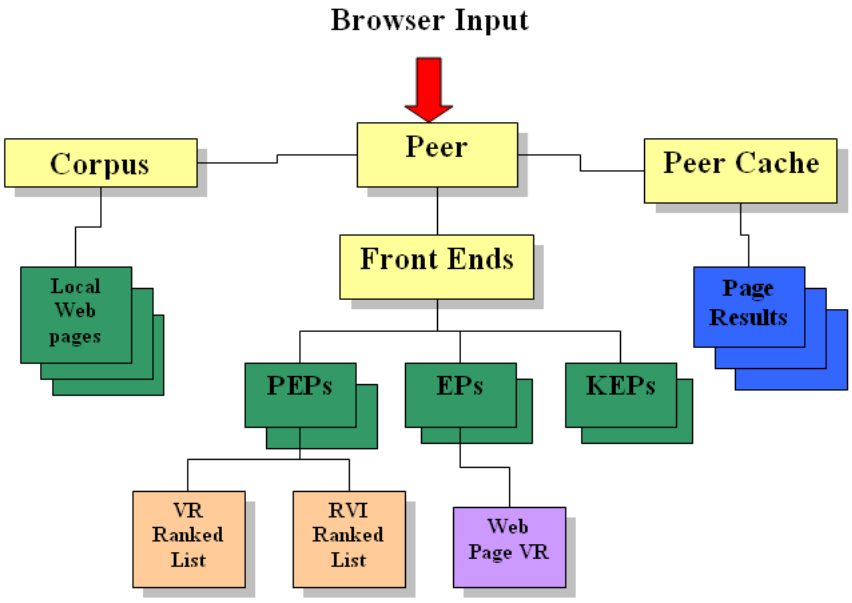

Figure 4.4: Elite's Architecture Overview

constitutes the most complex part of the system. It deals with providing the peer with several interfaces to the P2P network according to decisions made by the community as described in detail in section *3.4*. The last module is completely optional as the Peer could simply act solely as a Simple Peer without having to take additional KEP, EP or PEP responsibilities.

The aim of the following section is to describe in detail the object oriented architecture of this prototype as well as fundamental modeling concepts.

4.3.3 Module Description

4.3.3.1 Peer Cache & Corpus

The Peer Cache and Corpus are straight forward and do not need further explanations than those stated in the previous paragraph. It will be noted however that the corpus is the only entity in the system modelling Web pages as *Local Web Page* objects (described in the next paragraph). Among others, these objects are the most detailed representations of web pages in our system. This corpus is the main local repository consisting of the users browsing history. It enables the creation of a peer identity and

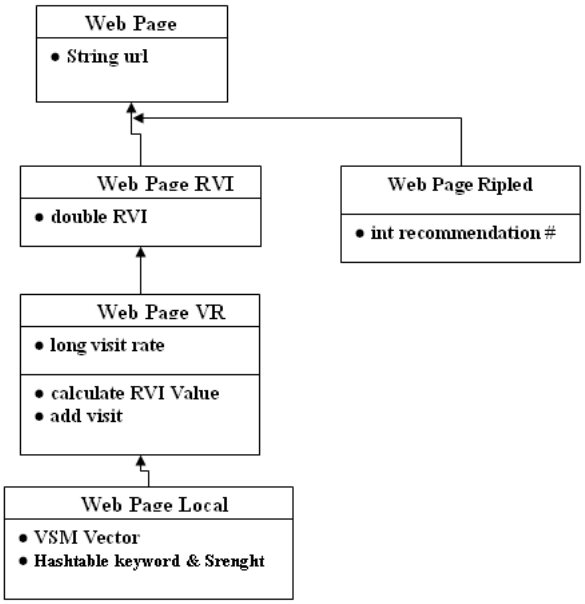

Figure 4.5: Web Pages

expertise which will then be used as the main criteria in joining elite groups.

4.3.3.2 Web Pages

Web pages could be seen as the main currency in *Elite*. These objects are constantly exchanged, accumulated and their wealth is the main asset of an individual Peer. *Elite* possesses 4 different types of model for Web pages depending on their usage. All of these models extend a very simple *Web Page* object, solely containing a String URL attribute which is used as a unique Id (figure 4.5).

The *Rippled Web page* object is probably the second simplest of all. It represents a page stored and recommended to Elite Peers. It directly extends the Web Page object and holds a recommendation_number integer representing the amount of times this Elite Peer received a ripple containing this page. This variable is then used to rank a set of Rippled pages whenever a query for the keyword of expertise is initiated by the user.

The *RVI Web Page* is an extension of the web page root object that takes into account the relative visit increase of such a page. These pages are stored and exchanged

by PEPs. They are the primary resources while creating the RVI Elite ranking result set.

A very similar class derived as a child to the latter is the *VR Web page*. It adds another layer of elite Web page modeling to its parent by representing the visit rate of pages in the system. These pages are stored by EPs and PEPs in the system. Page Elite Peers can, periodically, compute a page's global Relative Visit Increase so as to, update it's global entry in another PEP.

Finally, the *Local Web Page* object, as mentioned in the previous paragraph, is used by the corpus to store and compute a Peers expertise and interests. It inherits all of it's parents attributes and adds to them a Vector Space Model for each pages ever browsed. This vector is a sort of a web page summary describing it's main subjects and relative importance.

4.3.3.3 Ranked List

The next obvious model to explore thereafter is the ranking of these Web pages. Apart from the straight forward rippled ranking, *Elite* possesses 2 different types of ranking systems, both stored by PEPs, based on page's global Visit Rate or Relative Visit Increase. Both of these rankings extend the *Ranked List* object (figure 4.6) consisting of a hashtable with a maximum size of 10, using web pages URL's as Keys and either *RVI Web Page* or *VR Web page* as Values. Both of these ranked list possess methods enabling them to add or transfer pages from or to other PEPs. Each type of ranked list however possesses a different comparator object to arrange Web pages in the required order.

4.3.3.4 *Elite* & *Scribe* Messages

Messages in *Elite* are modeled very easily by extending either the *Elite Message* or *Scribe Message* object depending on the type desired. Both of these objects possess a type attribute integer assigned to each Elite or Scribe message as an identifier used when receiving them by Peers. An Elite Message additionally holds the *Pastry* source and destination Id needed to route the message through the overlay.

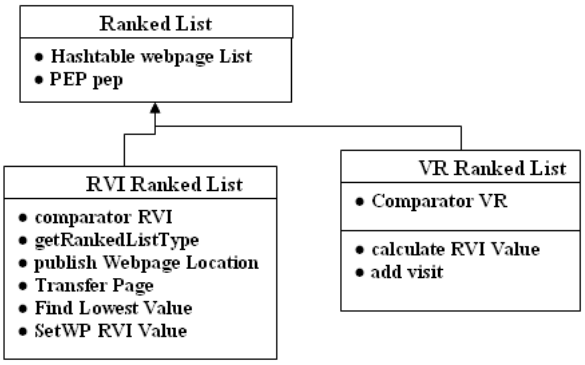

Figure 4.6: Ranked List

4.3.3.5 Peer

Peer modeling possesses a higher level of complexity than the models described previously. It is the main central entity in the system holding all modules together and therefore assuming the most important tasks in the overall structure. It's different functionalities are segregated on 6 different hierarchical levels (figure 4.7).

The first major functionality it is responsible for consists of joining the *Pastry* ring community from a network point of view embodied through the *Peer Ring Layer* Class. This class generates a *Pastry* Id assigned to the Peer for the rest of it's session by hashing it's IP network address.

The second important function held by the Peer is to manage several roles it might be assigned to. In other words it needs to deal with acting as KEPs, PEPs and EPs for potentially several keywords. Everything related to front ends is therefore located in the *Peer Front End* class, which extends the previous one.

Following this class in the hierarchy is the *PeerScribe* class, dealing without any surprises with all Scribe related issues. It therefore possesses Anycast, Deliver, Subscribe, Unsubscribe and Publish methods characteristic of any Scribe agent in a system as well as a list of STopics it is currently subscribed to. Although it possesses these functionalities, it never really uses them directly. These methods are called by the front ends seen previously. This layer also acts as a dispatcher, receiving Scribe messages and forwarding them to the proper front end.

Next are the Peer Elite Message Sender and Receiver classes which, as their

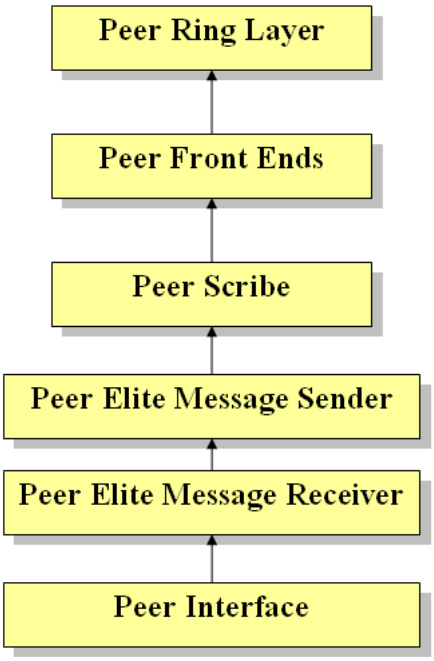

Figure 4.7: Peer Hierarchy

name suggest, are dedicated to handling the sending, forwarding and processing of any Elite message type possibly encountered. The latter class possesses a method for every type of messages which handles it accordingly.

Finally, the *Peer Interface* class is the most basic of all the ones mentioned previously. Public methods used externally are all located in this class. The visiting(String URL) method for example is called by the browser each time it browses a page, simply informing Elite of this event which deals with it accordingly (inserting new a Web page object if it wasn't aware of it or incrementing the corresponding Web Page objects number of visits).

4.3.3.6 Front Ends

The last quite complex structure to be described is that of Front Ends. Each type of front end possesses different needs, however they all share characteristics in common which are modeled in the *Front End* class (figure 4.8). Whether they are KEPs, EPs, or PEPs, all of them are "orbiting" around a keyword concept or Elite group. Hence, every

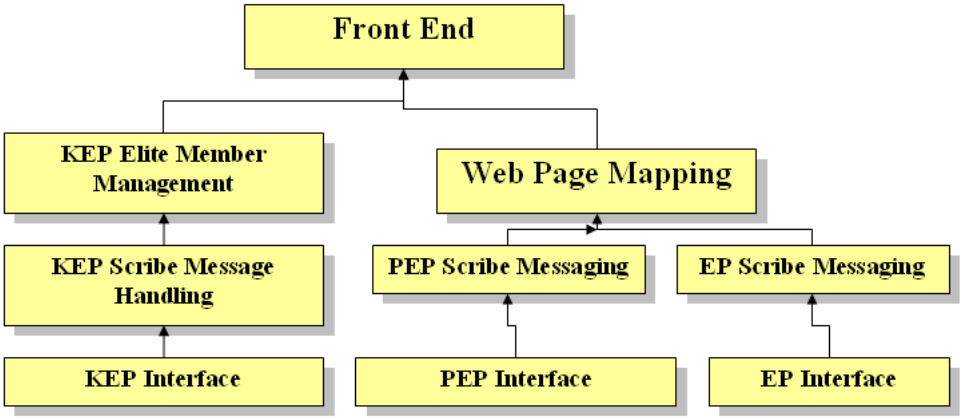

Figure 4.8: Front End Hierarchy

front end will possess a String representing the "keyword environment" it is assigned to as well as an integer holding the total number of corresponding PEPs which can possibly be queried for that keyword. Each front end will also hold a reference to the Peer it represents along with a list of Scribe Topics it is subscribed to.

4.3.3.7 KEPs

Keyword Elite Peers front ends are the simplest front ends to understand. They follow the same conceptualized functionality layering as does a Peer. A *KEP Elite Member Management* class for example deals with every issue concerning Elite group management and directly extends the Front End class. It keeps track of the minimum entry requirements, the number of Elite members currently in the group and decides to add or remove members according to decisions explained in the design Chapter of this document. It also can initiate LEP relief Tasks if a threshold time elapses without receiving any LEP messages from Elite Peers. Finally, it also possesses 2 additional layers very similar to the Peers layers, one dealing with Scribe messages processing while the other acts as the public KEP interface available to other members to communicate with this front end.

4.3.3.8 Web Mapping Front Ends

Both EPs and PEPs front end hold an additional layer of functionality in common. Recall from section 3.4, how both entities map their web page corpus with the relevant PEPs holding these pages as VR or RVI entries. The *Web Page Mapping* class therefore fulfills this functionality by keeping 2 hashtables storing Web pages URL as Keys and mapping PEP as Values as well as PEP and their current lowest value relationship. Both front ends need to subscribe to web page STopics to receive mapping updates as well as answering any relevant mapping request from other front ends in the system unaware of specific mappings.

Similar to the Peer, both front ends possess their own respective *Scribe Messaging* layer with one difference that unlike the former which only knows how to dispatch these messages to the correct front end, these entities possess the relevant processing methods for each type of Scribe message.

Finally, both front ends also possess their own public interface, serving the same purpose as the KEPs. PEPs for example, need to possess a instance of both VR and RVI ranked list (section 4.3.3.3) as well as their own specific PEP name (section 3.4.4.1). They also possess methods giving them the ability to transfer VR or RVI entries between each other as explained in section 3.4.4.4. EPs, on the other hand, need to know whether they are the LEP of the Elite group or not and therefore also need to know the Peers relevant expertise in their associated Keyword both modeled as attributes of their Interface class. Every EP additionally possesses a set of ripple pages received and sent and possess the ability to ripple pages they judge good enough as well as informing correct mapping PEPs of their visit to specific web pages.

Chapter 5

Evaluation

This chapter reports all of our results and findings gathered during this research. In the first section, a formal theoretical investigation of the *Ripple Effect* is performed followed by simulation results obtained for this technique. These findings suggest that the ripple technique reduces entrenchment effects and also improves general web browsing quality. Finally, a general evaluation of *Elite*'s design will be investigated in the last section.

5.1 Formal Analysis Evaluation

Previous to any simulation evaluation, a theoretical analysis and understanding of the *Ripple Effect* model imposes itself. This section therefore, analyses the most important equations derived in the previous chapter so as to acquire a general appreciation of this algorithm.

5.1.1 Probability of Incrementing Previous Recommendations

In figure 5.1, the probability of a ripple message incrementing a previously received recommendation, as opposed to hitting an unaware Elite Peer, according to the number of trials being launched when the ripple size $s = 1$ is depicted. This curve represents equation **??** and in other words, estimate the probability of a ripple *not* stopping at trial T. Each curve represents a ripple being initiated when a specific percentage of the Elite community already has done so for the same page.

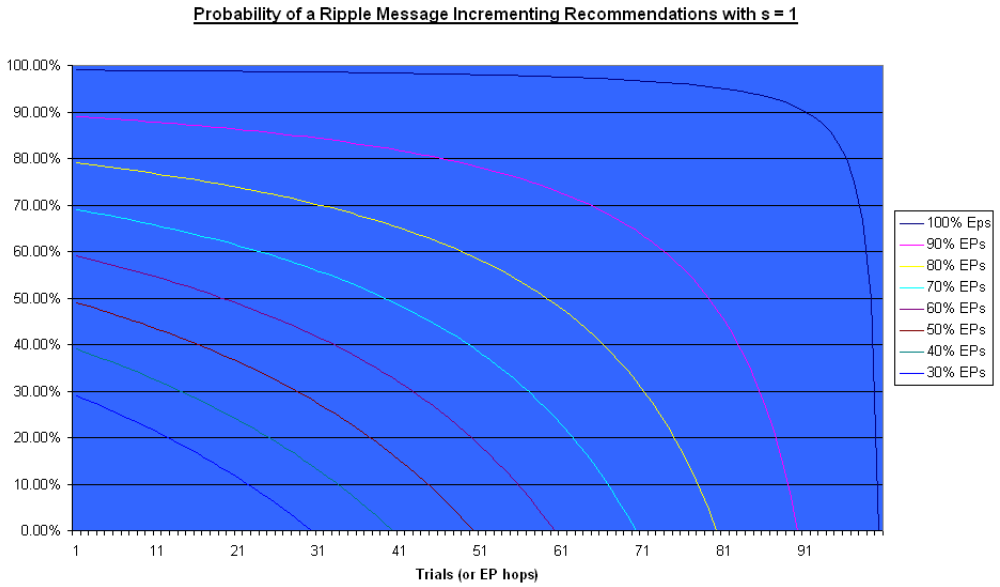

Figure 5.1: Probility of a Ripple Message Incrementing Recommendations with s=1

As you can see, it is clear that the more page p is being rippled, the more we can expect a ripple message to possess a large amount of trials. This is the behavior expected since, if only rippling is considered as a way for peers to become aware of a page, when a page is being rippled for the first time, no peer except the one initiating the ripple will be aware of it. Hence, when $R = 1$ (section 4.1), the first peer being hit by this ripple will automatically be unaware of this page, therefore the probability of hitting a previously recommended Peer at trial 1 will always be 0 for the first ripple. However, as the number of ripples being initiated increases, the number of peers which were previously recommended this page is equal to $s*(R-1)$. Therefore as R increases the probabilit of a ripple message hitting an already recommended peer increases as well.

Moreover, for any ripple, once the message is initiated, and since a ripple never encounters twice the same peer (section 3.7), the number of previously recommended peers in the group which haven't yet been hit decreases at each trial, hence the probability of hitting one as well.

Finally, when every Elite peer in the group except one has rippled page p, if this last peer decides to do so, we can see how the probability of this ripple having

an amount of trials close to the size of the elite group is high. This is simply because nearly every peer in the elite group was previously recommended this page, hence the algorithm will stop only when it meets the specific peer in the group which still hadn't been recommended this page. This represents a lot of messages, however, we need to keep in mind that this is only a theoretical description of the Ripple Effect, and the probability of every Elite peer rippling the same page is extremely close to 0 and can therefore be assumed never to happen.

5.1.2 Trial Probability

Following this observation, what we naturally wish to understand now is how the probability of having T trials evolves as the number of ripples for page p increases. Figure 5.2 shows a graph illustrating the latter. As expected, for $s = 1$, the probability of having at least 1 trial, for any ripple number, is always equal to 100% since in order for the algorithm to stop it needs to encounter one unaware peer. This graph directly links with the previous one since the probability of having a trial T is equal to the product of not hitting any unaware peer in the previous T-1 trials. This graph hence depicts equation 4.1.2.2. It is interesting to note how, for the first few ripples being initiated, the curve illustrating the probability of having a trial, ressemble that of a step function. And as the number of ripples increases, it becomes smoother to finally become totally linear when the last Elite peer in the group decides to ripple the page.

5.1.3 Probability of one particular Peer getting hit twice

Finally, the fundamental behavior we are interested in, consists of understanding the process by which a page will receive further recommendations and hence increase its rank in the rippled result set. Figure 5.3, graphs equation 4.12 which illustrates this behavior.

As you can see, a page will have a very low probability of getting hit twice in most cases, unless a large amount of peers decide to ripple the page. This is the behavior we wished to achieve since, the larger the number of peers rippling a page, the better we can assume its quality to be. Hence a page of high quality, or information valued by a large percentage of the community, will get its rank increased. In this case, the probability of a page increasing it's rank is very low when less than 70% of the

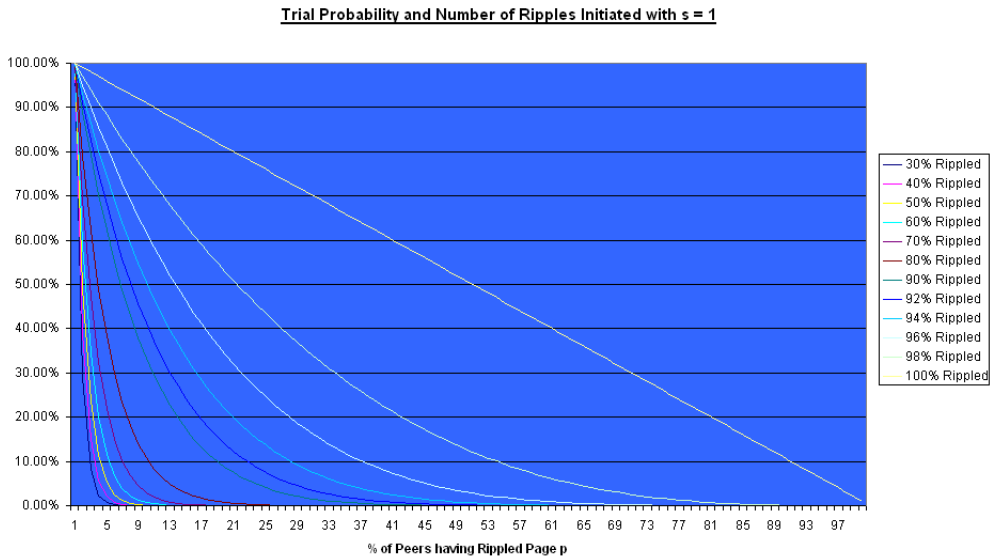

Figure 5.2: Trial Probability

population ripples it but then starts increasing exponentially when this threshold is reached. It can be argued, and this research agrees with it, that we should more than likely prefer a threshold set at a lower value. Even though this is probably the case, this graph nevertheless serves its purpose by proving how a community without any direct action, can passively promote information valued by the majority and therefore assign a higher rank to pages holding this data. Further investigations could research how variables could be used in order to set this threshold at a specific value.

5.2 Simulation Evaluation

As stated in section 4.2.1, once the *Ripple Effect* algorithm was mathematically formalised, a comparison as to how Web page awareness would evolve through time in both "ripple" and "no ripple" environments was made. Additionally, in order to tailor our Elite prototype and guarantee the most optimal results possible, we were interested in understanding what impact would ripple size s and minimum ripple requirements $RipMin$ criteria possess over the awareness evolution of pages. This section thus, outlines the results obtained during these experiments in details. A summary of all these findings can be found in appendix **??** as well as more detailed measurements used

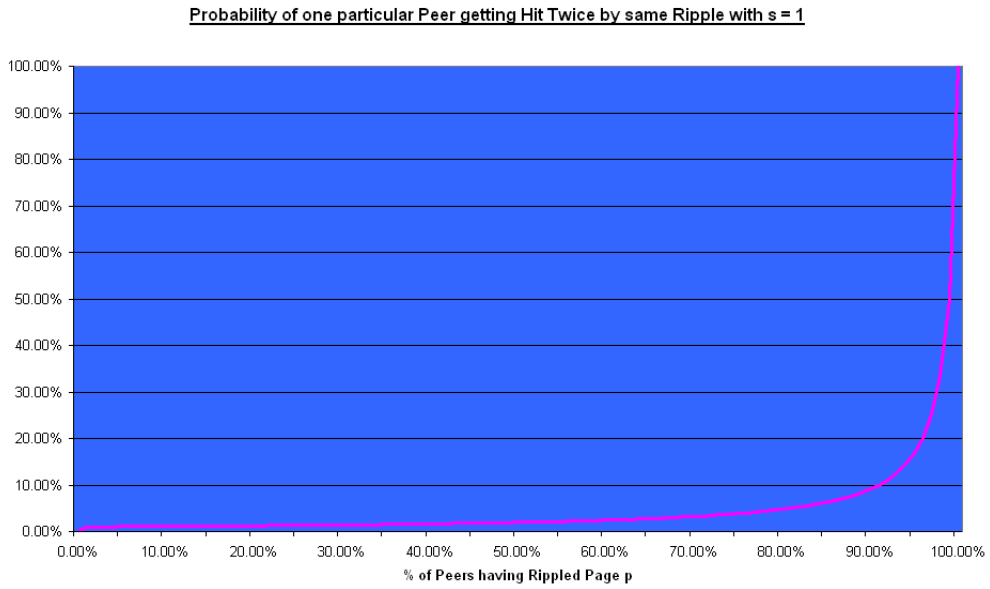

Figure 5.3: Probability of getting hit twice

to derive the graphs illustrated in this section.

All experiments were carried out using the parameters described in section 4.2.3 with a set of 200 Peers browsing a web consisting of 2000 pages periodically renewed (section 4.2.3) for a period corresponding to 5 years. While we are aware of the limitations of these experiments due to the relatively small number of Peers modelled, (owing to time & computational power available during these experiments), we do not expect however, that the variables measured are highly affected by this feature of our test bench and moreover, the results outlined below do suggest and describe interesting & genuine facts about our algorithm which we believe applies to any amount of peers. Our model nonetheless, assigned a ratio of 10 between the number of Peers and pages available which is the typical scenario investigated in such experiments. Nevertheless, further experiments investigating how the number of peers or even the ratio between peers and pages would influence our results would be interesting to pursue.

5.2.1 Pure Popularity Based Environments

As explained in the previous chapter (section 4.2.3), so as to maximise the quality of our comparisons between "ripple" and "no ripple" environments, we sought to take as a

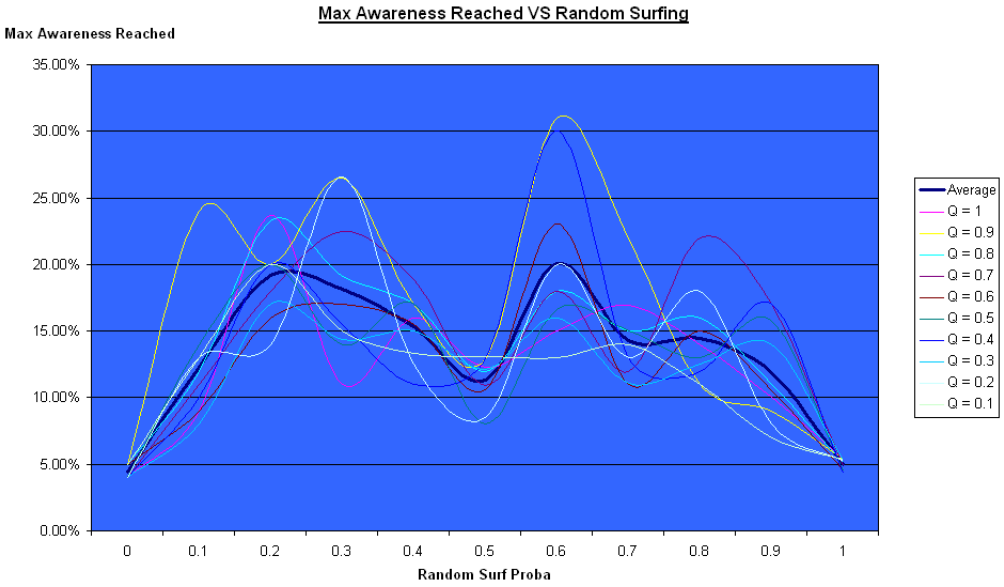

Figure 5.4: Maximum Awareness Reached in a "no ripple " environment

reference point, the best performance in the second case. Hence, a set of experiments were performed in order to find out for which value ψ (probability of a user doing random browsing) would a "no ripple" environment produce best results. All of the following ripple experiments will use this value so as to compare as accurately as possible both scenarios.(A notation summary is provided in table 4.1).

The first sets of experiments therefore consisted in running our model without any ripple initiated and varying ψ from a wide range of values ranging from 0 to 1. $\psi = 0$ would correspond to a situation where users would browse the web solely using a search engine to find new pages, while $\psi = 1$ represents the opposite when users never use search engines when surfing the Web.

Figure 5.4 outlines the maximum awareness reached by Web pages accross a set of ψ values. Each curve represents different quality pages $Q(p)$ ranging from 0 to 1. As you can see, surprisingly, the awareness distribution doesn't seem, at first glance, to be highly affected by the random browsing probability. Furthermore, quality seems totally unrelated to the maximum amount of awareness a page would achieve throughout its life time. Most of pages whether of quality 0.1 or 1 for example, reached a maximum awareness close to 20 % which is very low. An average curve was therefore graphed in

order to study the overall behaviour of a pure popularity driven search engine.

As you can see, pages in average perform equally bad in an environment consisting of pure popularity driven search queries as well as pure random browsing since the average pages would only reach 5% of awareness. Moreover, it can easilly be noticed that, although pages of different qualities have disparate values for most ψ's, they all converge to 5% for both $\psi = 1$ and $\psi = 0$. Additionnally, observing the average curve shows how page attain 2 slight peaks in awareness when $\psi = 0.2$ and $\psi = 0.6$ with the second achieving a better score. Interestingly, as pointed out in section 4.2.3, the value observed in real life for random browsing seems to assign ψ a value of 0.63 which is extremely close to the numbers outlined previously. It was therefore decided to use this value for all future ripple experiments as it both corresponds to a "no ripple" environment performing at its best and moreover it is very close to real life parameters. Finally, it must be pointed out that none of the pages in this scenario reached their TBP value which corresponds to the the time taken for a page to reach 90% of global popularity (section 4.2.1).

To summarise, this experiment therefore tells us 5 important things:

1. Page awareness in an environment driven by popularity search engines performs very poorly
2. Awareness doesn't seem affected by either page quality or random surfing probability
3. All pages however, perform very badly in pure search driven or pure random surfing environments
4. Although, performance is very low, 2 slight peaks seem to appear on average at values $\psi = 0.2$ and $\psi = 0.6$
5. Finally, no pages in the system was able to achieve its TBP value

5.2.2 Ripple Based Environments

5.2.2.1 TBPs & QCPs

Now that an analysis of a "no ripple" environment had been undertaken, a large set of experiments was performed. In each one of them, the same scenario as in 5.2.1 was

modelled, using this time, rippling of pages of different minimum quality thresholds, ranging from $Q(p) = 0.5$ to $Q(p) = 1$. Rippling pages of lower quality wasn't performed as we only seek to increase awareness of good pages in the system. Additionally, for each of those thresholds the rippling size s was also modulated within values starting from 1 to 5. As before, awareness evolution during time as well as TBP and QCP values (section 4.2.3) were measured.

The histogram in figure 5.5 shows the results for all these experiments. This graph illustrated the percentage of pages reaching their TBP value, in other words 90% of global awareness, throughout their life time. The x-axis represents the minimum quality required for a page to be rippled during the experiments where a value of 1.1 represents the same results when pages of quality equal or above $Q(p) = 1.1$ are rippled, (in other words no pages), corresponding to the findings in the previous experiment without any rippled being initiated at all. For each of the $RipMin$ values, the 5 different colored histograms correspond to different ripple size s tested. Finally, the two curves represent the QCP, or quality per click (section 4.2.3), metric for both "no ripple" and "ripple" environments.

The first obvious feature to point out is the difference in QCP value corresponding to both scenarios. While environments using no ripple achieved a stable quality per click of 0.5 in average, ripple environments always outperforms the mean browsing quality by a difference of 20% on average for $RipMin$ values ranging from 0.8 to 1. This value decreases as $RipMin$ is lowered down to 0.7 and stabilises at $QCP = 0.6$ which still is above a "no ripple" scenario. It is also worth mentioning that these values were equal for all ripple sizes s within each $RipMin$ scenario suggesting that ripple size isn't directly related to QCP.

Secondly, while no single page reached their TBP in "no ripple" environments, about 12% of them did in the latter cases. Moreover, no obvious recognisable identifiable pattern is observed when relating TBP performance and ripple size s values. In order to confirm this fact, an additional graph (figure 5.6) was plotted directly comparing the percentage of pages reaching their TBP in relation to ripple size. There again, ripple size seems to be unrelated to TBP performance.

Subsequently, the next question to ask is: amongs these pages, what is the average TBP value encountered? In other words how long does it take for a page in general to reach this TBP? Figure 5.7 shows this data plotted against RipMin values

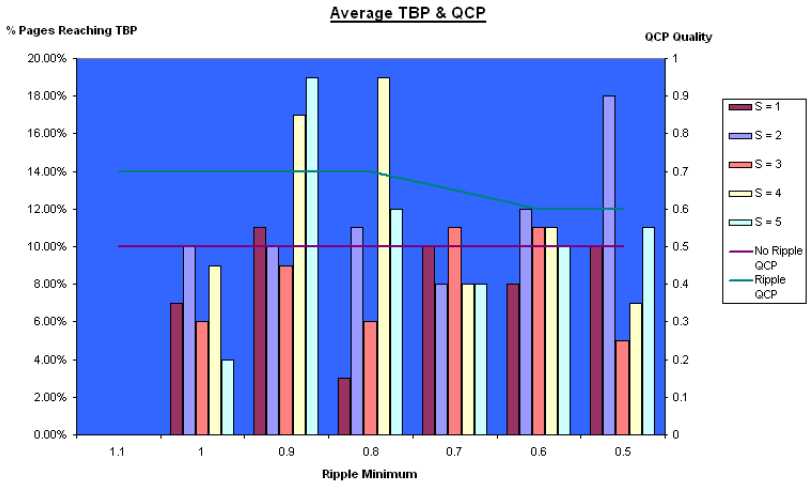

Figure 5.5: Average TBP & QCP

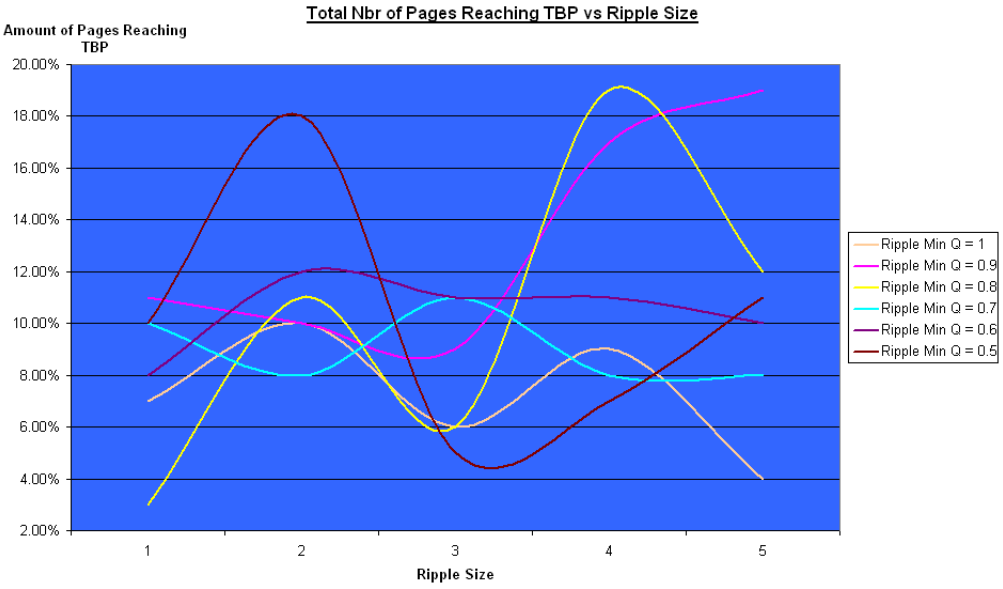

Figure 5.6: Pages reaching TBP and Ripple Size

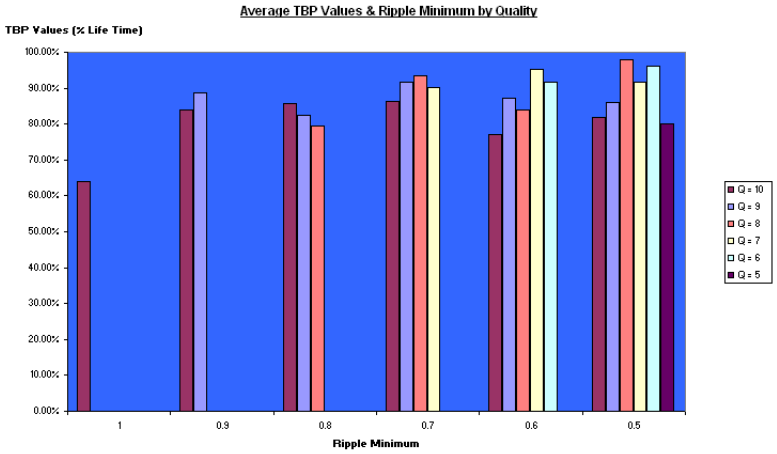

Figure 5.7: TBP Values and RipMin

for different page qualities. An interesting point to notice here, is the fact that only rippled pages reach their TBP. Put in another way, whether a page reaches its TBP is totally dependant on the $RipMin$ threshold used. When $RipMin = 1$, only pages of quality 1 reach TBP when $RipMin = 0.9$ only pages with $Q(p) = 1$ and $Q(p) = 0.9$ reach this value etc... Moreover, when solely pages of quality 1 are rippled, these pages reach TBP at 60% of their lifetime while as soon as pages of other qualities are added, the time taken for pages to reach TBP stablelises regardless of $RipMin$ around 90% which is quite late. Here again ripple size s doesn't seem to have an impact on these figures which is confirmed when plotting them directly together in figure 5.8.

Finally, the graph in figure 5.9 shows the percentage of pages reaching their TBP according to $RipMin$ values but this time for each page quality. Here again, notice how only the rippled pages reach their TBP but moreover, this graph shows how the benefits of the *Ripple Effect* are shared among pages while the threshold is lowered. When only pages of quality $Q(p) = 1$ are rippled, 100% of these reach their TBP which is excelent. However, as soon as $RipMin$ decreases so does this figure. When $RipMin = 0.9$, this number decreases by 30% and with $RipMin = 0.8$ it goes down to an average close to 37%. This value seems to stabilise around 20% for all other $RipMin$'s.

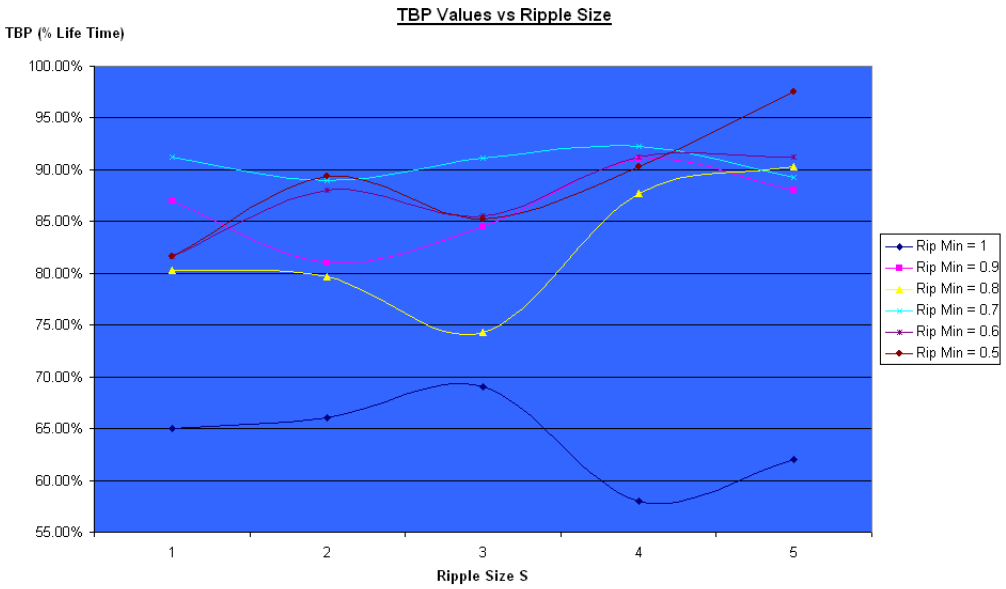

Figure 5.8: TBP Values and Ripple Size

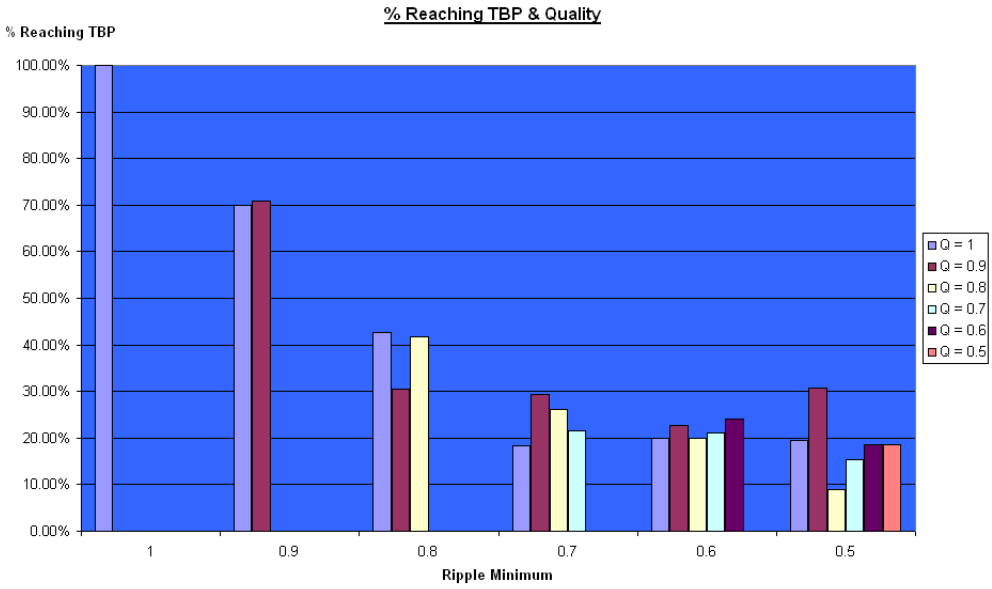

Figure 5.9: Percentage Reaching TBP and Quality

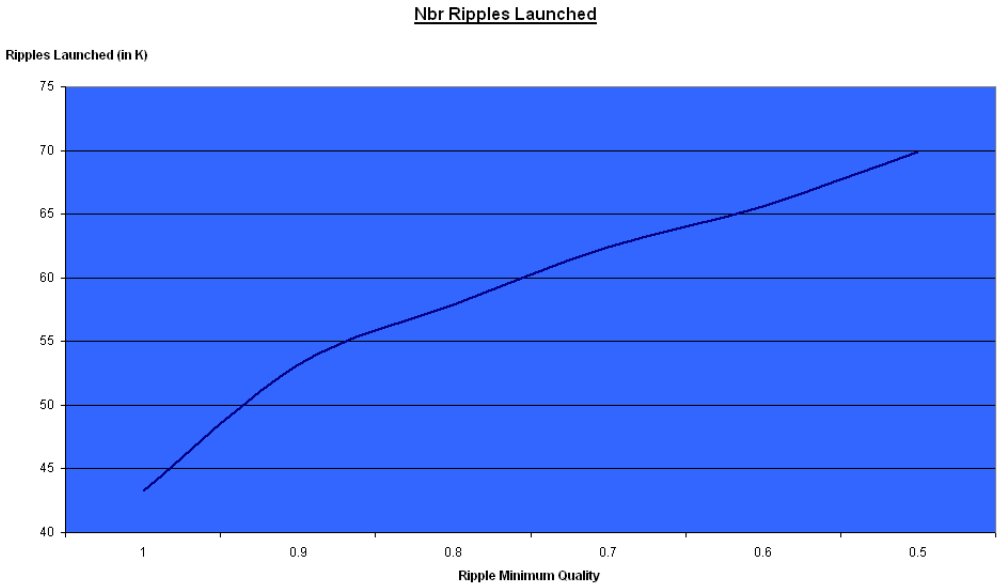

Figure 5.10: Number of Ripples Intiniated

5.2.2.2 Number of Messages and Ripples Initiated

During these experiments, the number of messages, or hops between peers, on average for 1 ripple, as well as the total amount of ripples initiated was measured in the environment. Figure 5.10 for example, depicts the number of ripples initiated according to $RipMin$ thresholds. As expected, when the ripple minimum quality requirements decreases, the number of ripples launched increases. This totally makes sense, since as $RipMin$ goes down, the number of pages available to ripple goes up, hence more pages are rippled.

The next graph (figure 5.11), shows the relation between the number of hops, or messages sent, per ripple in relation to the ripple size. This plot clearly shows that ripple size directly affects the number of hops on average executed per ripple. Again, these figures confirm what we expected since, as s increases, the number of previously unware Elite Peers to contact increases, hence the algorithm takes more time to stop and therefore hops along a larger amount of Peers.

Furthermore, it is interesting to notice how, when $s = 1$, $RipMin$ has no effect on these results while as s increases, the higher the minimum quality threshold, the higher the amount of messages sent. This is very surprising and totally unexpected

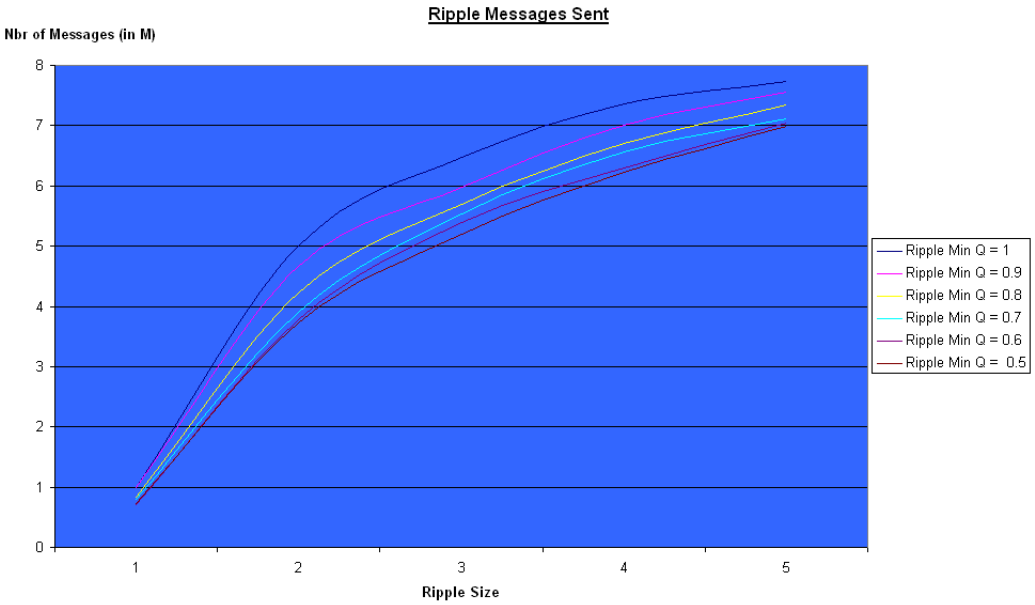

Figure 5.11: Ripple Messages Sent

from the author's perspective. With this discovery made, further investigations where needed and hence, an additional graph plotting the difference in messages sent between $RipMin = 5$ and all other $RipMin$ values was derived in figure 5.12. As you can see, the difference in the amount of messages sent is minimal when $s = 1$ as stated previously, it then reaches a peak at $s = 2$ for $RipMin >= 8$ and decreases again as s moves towards 5. Also, it seems like this peak is slightly delayed towards $s = 3$ when $RipMin < 8$. Plotting the same results for higher s values would be interesting as these curves seem to converge towards another point further in the graph.

However, an initial proposition attempting to understand such a behavior could possibly be formulated. As we have seen in figure 5.9, when a page is rippled it directly affects positively its increase in awareness in the population, and moreover, as $RipMin$ increases the percentage reaching TBP goes up. Hence, since these pages rapidly gain awareness, we can assume them to be ranked high in the VR ranking. Therefore an increasing number of Peers will discover this page through search queries as opposed to rippling. Consequently, even though a page might not have been rippled many times, more peers in these cases will already be aware of it from browsing or querying, hence a ripple will take more hops, or message, to end. Furthermore, figure 5.9 backs this

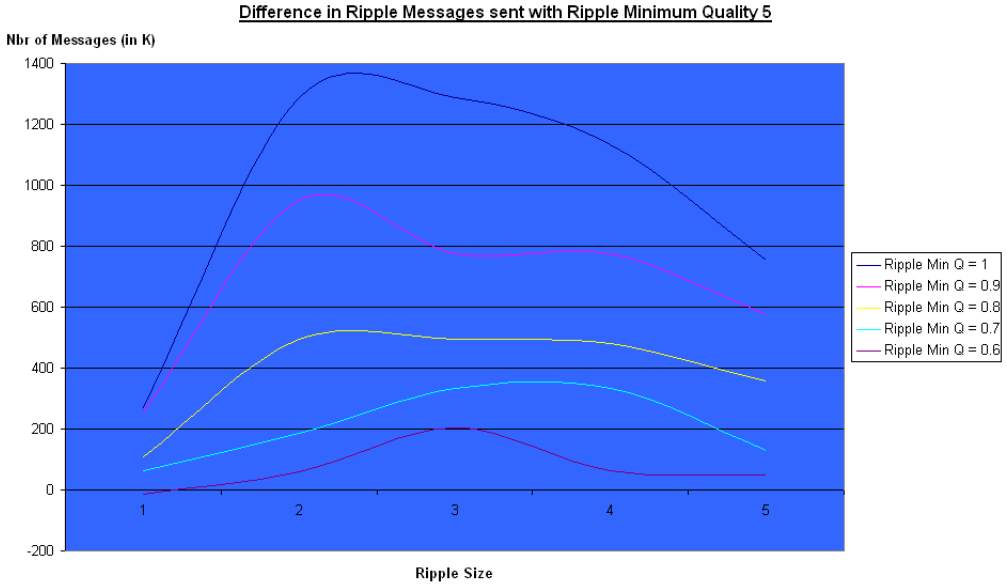

Figure 5.12: Dif in Ripple Mgs Sent with Ripple Size = 5

idea, since as *RipMin* decreases so does the percentage of pages reaching their TBP, and ultimatly so does the additional number of messages due to *RipMin* which is what is observed in figure 5.12.

Another set of graphs were derived so as to compare the number of messages sent between ripples of size $s > 1$ and ripples of size $s = 1$ initiated several times in order to hit the same number of unaware peers when $s > 1$. We would expect ripples with s=1 initiated several times to, in overall, produce more messages than ripples reaching the same number of unaware peers with higher s's. But figure 5.13 shows us the complete opposite. Although the difference is very minimal compared to thousands of messages sent, this graph shows us how rippling with $s = 1$ is more efficient than rippling with higher s's. An average of 50 more messages are sent when using higher ripples sizes. This number increases as s reaches 3.5 and then decreases for values reaching a ripple size close to 5. Once again, we can notice how *RipMin* affects the difference in messages sent negatively. This however, assuming the preceding findings could be interpreted, is expected since we have seen how Rippling with a high quality threshold increases the number of messages sent. Rippling with size $s = 1$ several times, simply accumulates these differences. Moreover, as shown in figure 5.12 the difference in messages sent is minimal for $s = 1$ which is confirmed here as well.

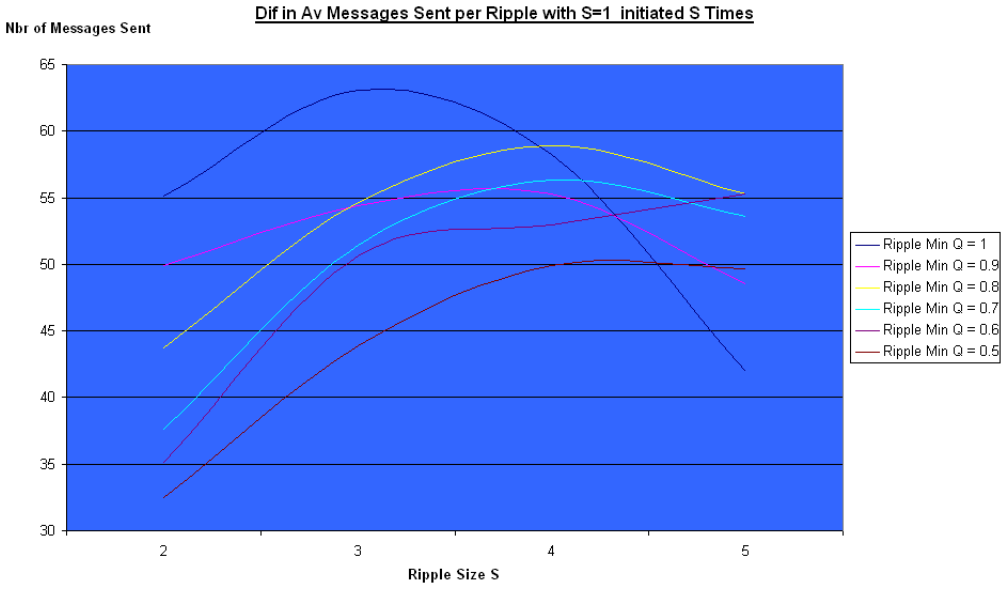

Figure 5.13: Dif in average mgs sent per Ripple with s=1 initiated s times

5.2.2.3 Variable Ripple Size

Finally, an additional case consisting of setting the ripple size s proportional to the quality of the page being rippled was tested as well. In this scenario, we set RipMin = 0.6, and assigned ripple sizes to pages of quality 0.7, 0.8, 0.9 and 1, respectively of values of 1, 2, 3, 4 and 5.

This experiment found that the percentage of pages reaching TBP as well as their TBP values was equivalent to previous experiments. The number of ripples initiated was similar to ripples with $RipMin = 6.5$ while the number of messages sent mapped those of previous ripples with a ripple size between 3 & 4. Finally, the QCP for such a scenario was found to be equal to 7.

5.3 *Elite* Design Evaluation

In this section, our design will be evaluated in detail, and particular emphasis will be given to the original objectives listed in the introduction (??). Each of these points will be examined in respect to the solutions outlined in previous chapters.

5.3.1 High Quality Results

Our first main objective intended implementing a search engine returning results of quality as high as possible. Chapter 3 outlined how selecting the most expert peers in the system for each specific subjects, enabled us to provide to the community this expertise shared by a group of individuals. Furthermore, section 3.4.2 illustrated how result quality was affected by the Elite's groups size and how the latter could be tailored in function of the communities needs. Furthermore, Elite's design was conceived so as to ensure the expert group dedicated in filtering pages was constantly maintained with the most expert peers in the community thanks to the LEP protocol (section 3.4.2.4). To summarise *Elite* guarantees a constant high quality of results to its users by using collaborative filtering of selected peers with highest interest in subjects queried.

5.3.2 Countering the Rich get Richer Phenomenon

A second important goal, consisted in reducing this rich get richer phenomenon (section 1.3.2.2) which is now known as being an intrinsic feature of the World Wide Web's nature. This document showed how because of their design based on link analysis (section 1.3.2), current search engines could only worsen this effect. *Elite* solves this problem by shifting it's conception of page quality from links to people (section 3.1) hence removing completely this side effect held by other search engines.

5.3.3 Minimising Entrenchment Effects

Entrenchment effects was another important feature to tackle in our system. In order to do so, we designed a *Ripple Effect* technique which democratically, artificially increases the awareness of newly created pages based on their quality (section 3.7). A formal model of this algorithm was derived (section 4.1), analysed and proved to possess the characteristics desired. Moreover, a simulation was carried out (section 4.2.1) in order to study the impact of such an algorithm in the community. The results showed how rippling newly created pages improved the general browsing quality and increased the global awareness of these pages (section 5.2) thus reducing entrenchment effects and therefore achieving our goal.

5.3.4 Information Control

Control over digital data was taken deeply into consideration when designing *Elite*. Thanks to a new GKD Indexing Scheme (section 3.4.4), malicious manipulation of result set ordering is made very hard if not impossible. Furthermore, *Elite*'s encryption infrastructure (section 3.6), adds an additional layer of complication for potential evil entities making any possible data manipulation pointless.

5.3.5 Scalability

Scalability was of course the most important concern in this design, as *Elite* can only provide a valuable service if it is spread globally. Different scalability aspects were taken into consideration.

Web scalability as described in section 1.3.1, was countered by using a different approach than traditional search engines. Instead of attempting to index as many pages as possible, *Elite* uses only a pre-filtered web indexing mechanism, "distilled" twice, and obtained through peer browsing analysis which only takes into account pages which people are interested in (section 3.3.1). Secondly, a garbage removal feature was taken into consideration as well (section 3.4.4.5) enabling any *Elite* system to adapt its Web index according to network capabilities.

Scalability was also considered from a more technical point of view with the existence of KEPs regulating the size of elite groups according to the network load and demands (section 3.4.2). Storage is reduced to a minimum since, no pages are actually stored in the system, solely their Id and subjects of interests. Caching was also incorporated in the system removing any bottlenecks on PEPs due to popular queries and autonomous behavior was guaranteed by enabling *Elite* to handle node failures automatically (section **??** & **??**).

Finally, scalibility from a more human perspective was taken into account when designing the recommendation system (section 3.7.1.2) based on the Ripple Effect. This system is a zero input mechanism which enables total independence of people's willingness to recommend pages and hence is applicable on a global scale.

5.3.6 Anonymity

Anonymity in *Elite* is ensured at numerous levels. It is encorporated in Elite group's membership thanks to the LEP protocol as well as temporal Elite Ids (section 3.4.2.4, **??**), guaranteeing the free provision of information by any party. And more important, keyword queries executed throughout the system are also totally anonymous thanks to forwarding and source Id substitution (section 3.5.3) protocol.

5.3.7 Censorship

Finally, and probably most importantly, fundamental rights were a major consideration in our design. A fair censorhip, based on universal values, was designed based on the *Ripple Effect* technique (section 3.7). This enables the community as a whole to decide what information should be provided to the community and be rewarded for their "altruistic" behavior. It also guarantees protection of fundamental rights such as privacy as well as insuring as much freedom of expression as possible through democratic web decisions.

5.3.8 Further Evaluation

Our prototype being implemented and fully tested, further evaluations based on this prototype would be interesting to investigate so as to learn what parameters such as rate of updates, size of elite groups, number of PEP a network could handle etc... should be used in a global *Elite* deployment.

Chapter 6

Conclusion

This chapter aims at summarizing this research's outcomes in tackling issues raised in the initial part of this document. It starts by summarizing in section 6.1 the design and implementation approaches undertaken, attempting to provide answers to engineering difficulties encountered in this field. Section 6.2, will give a concise description of the analysis carried out of the *Ripple Effect*, followed by an outline of this research's contributions and possible future investigations in section 6.3 and 6.4 respectively.

6.1 *Elite*, Design with an Ethical Perspective

As a result of this research, several major deficiencies present in contemporary search engines were underlined and an attempt to overcome these issues as far as possible was undertaken. We proved that engineering design could have an important impact on improving how our society will evolve in the future by providing more adequate tools to assist these transformations. Thereupon, a novel category of search engine was designed, built and tested based on design principles including strong ethical considerations.

This research showed how the "Rich get Richer" nature of the web for example, could be reduced by removing the dependency of search engines over links in determining importance of pages. The latter could instead be replaced by peer selection and collaboration based on expertise and feedback (3.1), simultaneously providing high quality results to users.

A new *Ripple Effect* technique built on top of *Pastry* was introduced and used in a zero input recommendation algorithm so as to counter entrenchment effects by artificially increasing awareness of new high quality pages. A deep analysis of this technique was performed (results summarized in the next section) which proved the effectiveness of this method.

Web scalability was dealt with by replacing exhaustive indexes with pre-filtered ones obtained via Web browsing instead of Web crawling. Regulating algorithms were additionally introduced to manage this index as needed (3.4.4.5) in respect to demand & network capabilities and the need to store indexed pages was removed.

Scalability from a technical point of view was taken into account by incorporating caches 3.5.4 to remove bottlenecks, minimizing messaging (3.4.4.3) and storage requirements.

Furthermore, anonymity was taken care of on numerous occasions throughout the design of this search engine, by guaranteeing confidentiality of message passing (3.5.3), anonymous group membership (section 3.4.2), group management protocols (LEP protocol 3.4.2.4) and hashing of identities (section 3.6) throughout the community.

Finally, a large amount of consideration was given to control over digital information. We ensured that global data indexing and associated responsibilities was fragmented and handed down to the people through a new GKD Indexing Scheme 3.4.4. This decision additionally assured limited damage of any potential index manipulation by malicious individuals which is made pointless thanks to a combined encryption mechanism 3.6. And most important, a censorship model, promoting universal values and protecting fundamental rights, was designed using democratic decision making by the community through the use of the *Ripple Effect* technique (section 3.7.2).

6.2 Ripple Effect Results

6.2.1 General Ripple Conclusions

A strong analysis of this new *Ripple Effect* technique was performed through formal and simulation investigations. The effectiveness of this method proved to reduce entrenchment effects of new high quality pages and increased general web

browsing quality. Optimal performance was observed for high minimum ripple quality requirements ($RipMin$) while efficient use of this technique was accounted for when the ripple size was set to 1. More precise results are outlined in the following section.

6.2.2 Detailed Ripple Results

Our results found that, rippling new pages could increase the quality of web browsing up to 20%. In our model, while no single page reached 90% of their maximum potential popularity in an environment without ripple effects, about 12% did in the alternative case. It was shown that the minimum quality requirements value used when rippling had a direct impact both on which pages reached high awareness as well as the time taken by these pages to do so. As the requirements are lowered, the benefits of rippling is shared among pages.

Also, we saw, to our surprise, how the minimum quality requirements had an influence on the number of hops or messages sent in an Elite group per ripple, and formulated an initial possible explanation for such a behavior (section 5.2.2.2).

Finally, our analysis also demonstrated how, initiating a ripple of size 1 several times as opposed to initiating only 1 with a larger size had no major impact on the difference of messages sent, but nevertheless, to our surprise, was to the benefits for ripples of size 1.

6.3 Contributions of this Research

This research initially aimed at raising awareness on critical issues faced by our society in relation to the increasing importance of search engines in the global community along with the responsibilities of engineers designing them. A detailed examination of these tools was performed so as to understand the technical difficulties associated with their design. We showed how a new approach to search engine implementation could attempt to answer these difficulties and achieve better ethical standards, thus proving how engineering design has a major impact on how our society could evolve in the future by providing better tools to the community. As a consequence, a new *Elite* search engine, based on Peer 2 Peer technology and collaborative peer filtering, was designed, built and tested, using a new *Ripple Effect* technique and GKD indexing

scheme approach, ultimately resulting in the creation of the *Elite Project* open source software freely available on the web.

6.4 Future Research Considerations

This initial project could potentially extend itself on a very wide scale at different levels. Additional research, for example, in analyzing the *Ripple Effect* technique would be interesting. The formal analysis performed in this document, for instance, only considered cases when ripple size was equal to 1. Investigating what influence would an increase in s have theoretically on each curve would be valuable. Further investigation within the simulation analysis is also required with a higher community of peer, so as to ensure, as we expect, that our results are consistent for any community size with a same ratio of associated pages. However, modulating this ratio would probably result in different figures which would be interesting to observe. Also, as stated in section 5.1.3, previous to any serious global deployment, an investigation in the variables shaping theoretical curves of this effect would be needed in order to obtain the correct curves required for information to flow via rippling on a global scale.

The most obvious next step in this project however, would involve improving the current implementation by adding the encryption and censorship mechanism outlined in (3) which would be a necessary requirement for full scale deployment. Improvements on the design could be also investigated based on suggestions outlined in 3.8. And finally probably the most exciting endeavor, would consist of incorporating *Elite* as a plugin application within most popular Internet browsers so as to enable a full deployment of this system and analyze its behavior on a global scale.

Bibliography

[1] Rice university. http://www.rice.edu/.

[2] Faroo home page. Accessed: 14/09/07. http://www.faroo.com/.

[3] Freenet. Accessed: 14/09/07. http://freenetproject.org/.

[4] Netbooster. Accessed: 14/09/07. http://www.netbooster.co.uk/.

[5] Open directory project. Accessed: 14/09/07. http://dmoz.org/about.html.

[6] R. Akavipat, L.-S. Wu, F. Menczer, and A.G. Maguitman. Emerging semantic communities in peer web search. 2006.

[7] Charles Reis Paul Willmann Peter Druschel Dan S. Wallach Xavier Bonnaire Pierre Sens Jean-Michel Busca Alan Mislove, Ansley Post and Luciana Arantes-Bezerra. Post: A secure, resilient, cooperative messaging system.

[8] Ntoulas Alexandros, Cho Junghoo, and Olston Christopher. What's new on the web?: the evolution of the web from a search engine perspective, 2004. 988674 1-12.

[9] Broder Andrei, Kumar Ravi, Maghoul Farzin, Raghavan Prabhakar, Rajagopalan Sridhar, Stata Raymie, Tomkins Andrew, and Wiener Janet. Graph structure in the web, 2000. 346290 309-320.

[10] Miguel Castro Antony Rowstron, Anne-Marie Kermarrec and Peter Druschel. Scribe: The design of a large-scale event notification infrastructure? 2001.

[11] Peter Druschel Antony Rowstron. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. Oct 2001.

[12] Arasu Arvind, Cho Junghoo, Garcia-Molina Hector, Paepcke Andreas, and Raghavan Sriram. Searching the web. *ACM Trans. Inter. Tech.*, 1(1):2–43, 2001.

[13] Mugizi Robert Rwebangira Avrim Blum, T-H. Hubert Chan. A random-surfer web-graph model.

[14] BERG A. CHIEN S. BAR-YOSSEF, Z. and WEITZ. Approximating aggregate queries about web pages via random walks. 2000.

[15] Stephen Barrett. Gridle: A peer web searching system. 2004.

[16] Mayank Bawa, Gurmeet Singh Manku, and Prabhakar Raghavan. Sets : Search enhanced by topic segmentation.

[17] BBC. Google hit by link bomber. Accessed: 14/09/07. http://news.bbc.co.uk/2/hi/science/nature/1868395.stm.

[18] BBC. Improving your rank. Accessed: 14/09/07. http://news.bbc.co.uk/2/hi/business/936200.stm.

[19] BBC. Miserable failure links to bush. Accessed: 14/09/07. http://news.bbc.co.uk/2/hi/americas/3298443.stm.

[20] BBC. Web search engine could do better. Accessed: 14/09/07. http://news.bbc.co.uk/2/hi/science/nature/388394.stm.

[21] BBC. Wmd spoof is internet hit. Accessed: 14/09/07. http://news.bbc.co.uk.

[22] John Kubiatowicz Ben Y. Zhao and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. April 2001.

[23] Google Blogspot. How much data does google store? Accessed: 14/09/07. http://googlesystem.blogspot.com.

[24] Miguel Castro. Peer-to-peer overlays: Structured, unstructured or both?

[25] UNESCO Chris Conley, Mary Rundle. Ethical implications of emerging technologies. 2007.

[26] Fox Christopher. A stop list for general text. *SIGIR Forum*, 24(1-2):19–21.

[27] Ian Clarke. Philosophy behind freenet. Accessed: 14/09/07. http://freenetproject.org/philosophy.html.

[28] Fagin. Combining fuzzy information: an overview. *SIGMOD Record*, June 2002.

[29] Electronic Frontier Fondation. Defending freedom in the digital world. Accessed: 14/09/07. http://www.eff.org.

[30] Gartner. Hype curve. Accessed: 14/09/07. http://www.gartner.com.

[31] S. L. Gerhart. Do web search engines suppress controversy? Accessed: 14/09/07. http://www.firstmonday.org.

[32] Google. Google's corporate philosohhy. Accessed: 14/09/07. http://www.google.com/corporate/tenthings.html.

[33] Garrett Hardin. The tragedy of the commons. *Science Magazine*, 1968.

[34] David Karger M. Frans Kaashoek Hari Balakrishnany Ion Stoica, Robert Morris. Chord: A scalable peer-to-peer lookup service for internet applications. 2001.

[35] Hai Jin, Xiaomin Ning, Hanhua Chen, and Zuoning Yin. Efficient query routing for information retrieval in semantic overlays. 2006.

[36] Thorsten Joachims. Optimizing search engines using clickthrough data. 2002.

[37] Matthew Hindmany Kostas Tsioutsiouliklisz Judy A. Johnsonx. Googlearchy": How a few heavily-linked sites dominate politics on the web. 2003.

[38] O'Madadhain Joshua and Smyth Padhraic. Eventrank: a framework for ranking time-varying networks, 2005. 1134273 9-16.

[39] Gerhard Weikum Josiane Xavier Parreira, Sebastian Michel. p2pdating: Real life inspired semantic overlay networks for web search. 2007.

[40] Cho Junghoo and Roy Sourashis. Impact of search engines on page popularity, 2004. 988676 20-29.

[41] Cho Junghoo, Roy Sourashis, and E. Adams Robert. Page quality: in search of an unbiased web ranking, 2005. 1066220 551-562.

[42] S. LAWRENCE and GILES. Accessibility of information on the web. *Nature*, 1999.

[43] Killian Levacher. The elite project open source search engine. Accessed: 14/09/07. http://sourceforge.net/projects/elite/.

[44] Anne-Marie Kermarrec Miguel Castro, Peter Druschel and Antony Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE JOURNAL*, 2002.

[45] Anne-Marie Kermarrec Animesh Nandi Miguel Castro, Peter Druschel and Atul Singh Antony Rowstron. Splitstream: High-bandwidth content distribution in cooperative environments. Feb 2003.

[46] Anne-Marie Kermarrec Antony Rowstron Miguel Castro, Peter Druschel. One ring to rule them all: Service discovery and binding in structured peertopeer overlay networks.

[47] Ronny Lempel; Shlomo Moran. Predictive caching and prefetching of query results in search engines. 2003.

[48] United Nations;. Universal declaration of human rights. 1948.

[49] S. Olsen. Does search engine's power threaten web's independence?

[50] Sergey; Motwani-Rajeev; Winograd Terry Page, Lawrence; Brin. The pagerank citation ranking: Bringing order to the web. 1998.

[51] S. Yu Philip, Li Xin, and Liu Bing. On the temporal dimension of search, 2004. 1013519 448-449.

[52] Microsoft Research. Pastry home page. Accessed: 14/09/07. http://research.microsoft.com/ antr/Pastry/.

[53] P. Druschel Rowstron and A. Past: A large-scale, persistent peer-to-peer storage utility. May 2001.

[54] O. D. Sahin, A. Gulbeden, F. Emekci, D. Agrawal, and A. El Abbadi. Prism: indexing multi-dimensional data in p2p networks using reference vectors. 2005.

[55] Ricardo Baeza-Yates Carlos Castillo Felipe Saint-Jean. Web dynamics, structure, and page quality. 2002.

[56] Pandey Sandeep, Roy Sourashis, Olston Christopher, Cho Junghoo, and Chakrabarti Soumen. Shuffling a stacked deck: the case for partially randomized ranking of search engine results, 2005. 1083683 781-792.

[57] Peter Druschel Sitaram Iyer, Antony Rowstron. Squirrel: A decentralized peertopeer web cache. 2002.

[58] United States of America Department of State. Global internet freedom task force (gift). Accessed: 14/09/07. http://www.state.gov/r/pa/prs/ps/2007/79486.htm.

[59] Click Stream. Click stream study. Accessed: 14/09/07. http://www.websiteoptimization.com/speed/tweak/clickstream/.

[60] Yinglin Sun, Liang Sun, Xiaohui Huang, and Yu Lin. Resource discovery in locality-aware group-based semantic overlay of peer-to-peer networks. 2006.

[61] Chunqiang Tang, Zhichen Xu, and Sandhya Dwarkadas. Peer-to-peer information retrieval using self-organizing semantic overlay networks. 2003.

[62] International Herald Tribune. China defends internet censorship. Accessed: 14/09/07. http://www.iht.com/articles/2006/02/14/business/net.php.

[63] International Herald Tribune. Republicans hit in 'google bombing'. Accessed: 14/09/07. http://www.iht.com/articles/2006/10/26/business/google.php.

[64] UNESCO. Code of ethics for the information society. Accessed: 14/09/07. http://portal.unesco.org.

[65] UNESCO. Constitution of the united nations educational, scientific, and cultural organization. Accessed: 14/09/07. http://www.icomos.org.

[66] UNESCO. E governance. Accessed: 14/09/07. http://portal.unesco.org.

[67] UNESCO. Internet governance conference 2006. Accessed: 14/09/07. http://www.igfgreece2006.gr/.

[68] Rice University. Free pastry. Accessed: 14/09/07. http://freepastry.org/.

[69] Peter Lyman Hal R. Varian. How much information. 2003.

[70] Kandinsky Wassily, editor. *Point and Line to Plane*. Courier Dover, 1979.

[71] Le-Shin Wu, Ruj Akavipat, and Filippo Menczer. Adaptive query routing in peer web search. 2005.

[72] Torsten Suel Chandan Mathur Jo-WenWu Jiangong Zhang and Alex Delis Mehdi Kharrazi Xiaohui Long Kulesh Shanmugasundaram. Odissea: A peer-to-peer architecture for scalableweb search and information retrieval. 2003.

Appendix A

Notation & Concepts Summary

A.1 Acronyms

Acronym	Signification
KEP	Keyword Elite Peer
VR	Visit Rate
RVI	Relative Visit Increase
EP	Elite Peer
PEP	Page Elite Peer
LEP	Least Expert Peer
STopic	Scribe Topic
TBP	Time to Become Popular

Table A.1: Acronyms used throughout this document

A.2 Elite Messages

A.2.1 Pastry Messages

Message	Acronym
Join Request Message	JRM
Elite Member Status Message	EMSM
'Elite Peer Alive Message	EPAM
VR Update	/
RVI Update	/
Transfer VR Webpage Message	TVWM
Transfer RVI Webpage Message	TRWM
Query Answer Message	QAM
Censorship Alert Message	CAM

Table A.2: Pastry Messages

A.2.2 Scribe Messages

Messages	Acronym
Webpage Location Update Message	WLUM
Webpage Location Request Message	WLRM
Lowest PEP Value Message	LPVM
Ripple Webpage Message	RWM
Support Censorship Alert Message	SCAM
Censorhip Decision Message	CDM

Table A.3: SCRIBE Messages

A.3 Symbols

Symbol	Meaning
K_1, K_2,...	Keyword1, Keyword2,...
W_1,W_2,...	Webpage1, Webpage2,...
s	Ripple Size
S	Solution variable used to decrypt $H(url)$
P(p,t)	Popularity of page p at time t
V(p,t)	Visit rate of page p at time t
Q(p)	Quality of page p
A(p,t)	Awareness of page p at time t
I(p,t)	Increase in Popularity of page p at time t

Table A.4: List of symbols used throughout this document

A.4 Concepts & Definitions

Concept	Definition
Expertise and Interest	Used alternatively throughout document to refer to amount of browsing of a Peer in a given subject
Term	A word contained in a web page
Subject	Concept of interest to Peers associated with a given Term or Keyword
Keyword	A term specified in a Query or representing an Elite Group
Peer	A node in the pastry network, which is or isn't necessarily an EP, KEP or PEPThe word Peer is used to represent characteristics shared by every member of the network
High Profile Page	A page with many inward Links
Simple Peer	A peer which neither is a EP, PEP or KEP

Table A.5: Concepts and their definitions used throughout this document