GRAFT: A Distributed Recommendation Framework

by

Ferry Hendrikx

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2015

Abstract

Since the earliest human communities, reputation has been used by people to decide whether they should trust and interact with someone else. Traditionally, reputation was established through a person's standing, word of mouth and their associations. However, with the increasingly widespread use of the Internet, this situation has changed. In particular, all of the normal cues that help to build reputation are missing. Even the concept of identity is blurred by the common usage of pseudonyms.

In answer to this problem, many websites on the Internet have developed reputation systems that allow members to leave feedback about the performance of others in the execution of their duties. This accumulation of feedback about any individual can be used to characterise and predict their future behaviour in that context, allowing others to decide if they want to interact with that individual. Unfortunately, the information in each instance is limited to the narrow context of the website in which it was generated.

Not only is the reputation information constrained in context, it also limits the potential scope of what can be determined about an individual. The information that could be collected about entities includes social, demographic and reputation-based information. These are collectively called recommendation information in this thesis. Collecting this recommendation information from multiple sources and contexts should provide a wider view by which an entity can be evaluated than reputation alone could produce. The combination of these multiple sources of recommendation information can be naturally extended in the development of novel applications in areas such as access control and web service composition.

The GRAFT framework developed in this thesis encapsulates a paradigm shift in the way that reputation information is handled. It directly supports the collection and distribution goals by building a global distributed recommendation system that can be used to collect and make available recommendation information about both people and electronic services. This system can be used as both a drop-in replacement for existing systems, or it can be used to drive the consumption of recommendation information in novel new systems.

Recommendation information can be collected from both traditional reputation sources such as Amazon and eBay, and non-traditional reputation sources such as social networks, providing flexibility in what can be collected and subsequently utilised by consumers. The derivation of reputation information from non-reputation sources including demographic and social information, and the subsequent ability to use this recommendation information in the description and evaluation of policies is unique to GRAFT.

The major contributions of this thesis in the areas of reputation and reputation systems include the development of a reputation terminology, generalised models of reputation and reputation context, an extensive survey and taxonomy of reputation systems and a classification of existing reputation systems based on the taxonomy. This thesis also contributes an architecture for GRAFT, a prototype implementation of GRAFT showing its usefulness, and an evaluation that includes the results of a large number of simulation experiments showing how the architecture scales and handles both malicious peers and churn.

Acknowledgments

This thesis has only been possible with the support of numerous people. I am indebted to my supervisor Kris Bubendorfer for his insight and guidance. I am deeply grateful to my wife Radha, who supported me over countless weekends and nights, proofread every draft of this thesis, provided advice and brought clarity with her editorial expertise.

I would like to thank my examiners, Aaron Chen, Chris Scogings and Peter Komisarczuk, for their feedback and questions which have certainly strengthened this thesis. I would also like to thank Ian Welch for proofreading this thesis just before Christmas and for his advice; John Hine and Craig Wills, for supporting my application into the PhD programme; Simon Richardson and Martyn Bain, for many work opportunities (so we could pay for all of this).

Finally, my deepest appreciation to my parents, whose positive emphasis on education set the stage for my adult life; to the many friends who have been there for me throughout this journey; and especially to my loving children (James, Sienna, Anais and Niamh) who will no longer have to wait for dad to complete just one more paragraph before going for a swim or helping with your homework.

My sincere thanks to you all. Without your encouragement and support, none of this would have been possible.

Hendrikx

"Damnum appellandum est cum mala fama lucrum."
"Profit made at the expense of reputation should be called a loss."

- Publilius Syrus, fl. 46-29 BC.

"Associate yourself with Men of good Quality if you Esteem your own Reputation; for 'tis better to be alone than in bad Company."

- George Washington,
Rules of Civility & Decent Behavior in Company & Conversation, ca 1744.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Since the earliest human communities, reputation has been used by people to decide whether they should trust and interact with someone else. Reputation was built up and maintained by a person's standing and participation in their local community, their family and friends, and their participation at a place of worship [261].

In contrast, retailers relied on their physical presence, longevity of their business, and word-of-mouth [261, 159, 262] for their reputation. As most retailers usually operated only in the local market, customers could inspect any goods before purchase [262]. This requires little or no trust between the parties, as poor goods would simply not be purchased. Further, routine interactions with the same local retailers meant that the customers would soon know who was trustworthy, and who was not [262].

However, with the advent of the Internet, this situation has changed. The Internet provides for cheap, fast and easily accessible communications on a global scale. It is now possible for a single person to have contact with thousands of other people, potentially from all over the world. These contacts could be either local or remote, since the factor of distance is effectively removed. Given the millions of people online [263, 262], it is likely that a single interaction between two participants is both their first and last [88, 261].

In this electronic environment, the normal cues that help to build and establish reputation, such as social standing and word-of-mouth, are missing [88, 159]. Parties meeting on the Internet generally have no easy way to determine whether they should trust each other, or even carry out a transaction. Pseudonyms compound this problem further, especially considering they often change from website to website [112]. For example, a seller on an online marketplace may have a completely different name on another website, and it is possible that neither of these names is linked to a real world identity.

In answer to this problem, many websites on the Internet have developed reputation systems [261, 263, 104]. Typically, members of the website leave some kind of feedback about their transactions with other members. This feedback may be in the form of a numerical rating, a categorisation such as *good* or *bad*[263, 327], a vote, or a connection between members. Online auction sites are possibly the best known examples of these, but there are others. For example, Amazon[2] and Epinions[7] both use reputation systems to promote and sell products. These systems allow members of the website to build up a reputation that can be viewed by others.

Typically, the reputation of a single member consists of nothing more than an aggregation of all trades or interactions (both those with good and bad outcomes) that the member has had on the website [222]. In essence, members are able to decide whom they will trust based on past feedback. With a reputation system, members are more likely to be honest and well-behaved, as they have an expectation that current behaviour will impact on their future [88]. Since we are not able to build a long-term relationship with all other members, we instead rely on the feedback from a series of transactions that other members have had with the member in question. This series of transactions approximates some of the attributes of a long-term relationship [261].

Members with a better reputation are more likely to be trustworthy; they have a better track record, and have more to lose should a transaction go badly. In the case of eBay, the seller does not take on much risk because if they do not get paid, they do not ship the goods. However, the buyer must place some level of trust in the seller, by accepting the 'risk of prior performance' [159]. A reputation system ensures that the buyer can be confident about what he or she is going to buy if the seller has a decent reputation, and can therefore be trusted. As a result, good reputations can lead to more sales, and higher potential values for goods being sold [112, 263, 177].

In short, the reputation systems allow us to trust strangers by characterising and predicting [89, 262, 270] their future actions, based on their past actions. These sites would not be able to function without their reputation systems. Take eBay [6] as an example. It has over 4 million active auctions at any time [88, 261, 98]. The majority of the auctions on eBay are successfully transacted. According to Dellarocas [88] over 89% of the transactions on eBay are one-time only trades. eBay simply would not function if it did not have a reputation system.

This thesis takes a slightly wider view of reputation, considering all endorsements (good, neutral and bad) of a target entity to be useful in determining that entity's likely future behaviour. These endorsements are called recommendations and could be generated from a target's reputation, but also from their affiliations, capabilities and characteristics. However, discussions about the use of recommendations in this thesis will usually also refer to reputation.

GRAFT, or Generalised Recommendation Framework, supports the collection and distribution of recommendation information for both people and electronic services. The recommendation information is obtained from a variety of traditional and non-traditional reputation sources, and distributed in structured documents across a peer-to-peer (P2P) network to remove centralised bottlenecks, and ensure robustness of the overall system. The remainder of this chapter presents an overview of GRAFT, outlines the research goals, describes the contributions and scope, and finally, outlines the structure of this thesis.

1.2 Overview of GRAFT

GRAFT is a distributed framework that enables the collection and distribution of recommendation information about entities. These entities can be either people, or electronic services such as web services.

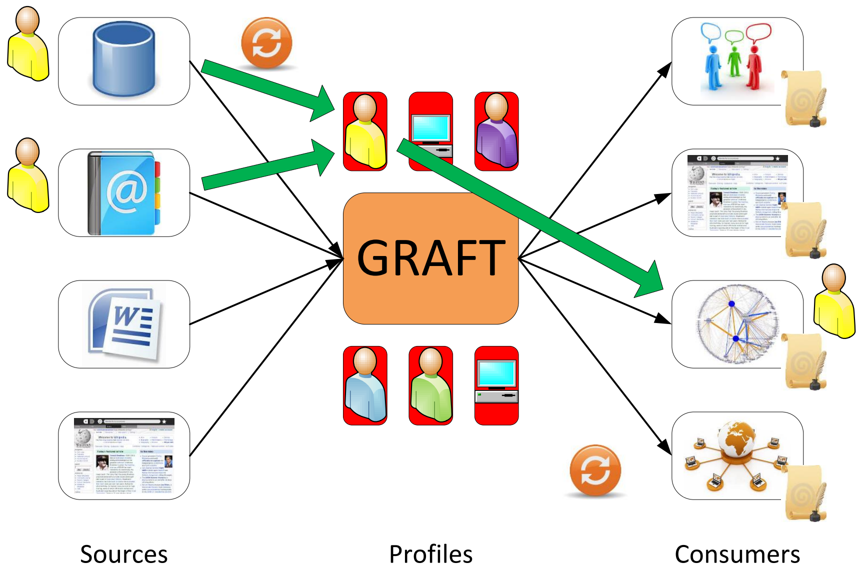

Figure 1.1: Overview of GRAFT, showing how multiple sources continuously feed recommendation information into GRAFT, and how this is continuously made available to consumers.

As shown in Figure 1.1, multiple sources continuously feed recommendation information into GRAFT, and this is continuously made available to consumers. Recommendation information in GRAFT is stored in structured documents called profiles. Every profile holds one or more recommendations, each from a unique source.

Every source of recommendation information has either explicit, or implicit reputation information. The feedback about individuals on eBay, for example, are a form of explicit reputation information, while the networks of professional colleagues formed on

LinkedIn [10] are an example of implicit reputation.

The consumers utilise the information obtained from the sources to make decisions about individual entities. For example, access control decisions made by a recommendation consumer about an individual might be influenced by the recommendations held by that individual. The exact policy implemented at each consumer is unique and only limited by the information made available by all of the sources.

All of the sources and consumers are nodes in a single, underlying peer-to-peer network. As part of its membership commitment, each node may be asked to store a number of profiles for GRAFT. Rather than store each profile in the network only once, they are each replicated multiple times, across many different nodes. Each profile is always tied to the identity of the entity it describes. This ensures that the profile can be located again, and that the profile is always available, even as other nodes join, or leave the network. Profiles are tied to individual entities using OpenID. All entities within GRAFT therefore have an OpenID, which acts as both the entity's identity, and as a key to locate their recommendation profile.

1.3 Research Goals

This thesis encapsulates a new paradigm for reputation that combines the derivation of reputation from multiple sources, including non-reputation sources, the integration of identity and reputation, and the integration of both human and electronic services into one framework. In particular, it examines different types of information that can be used as stand-ins for reputation, called recommendations, and the utilisation of these recommendations in the description and evaluation of policies. To that end, a number of research gaps were derived from the construction and subsequent analysis of the reputation systems taxonomy in Section 2.4.5. A subset of these gaps were selected as the research goals for this thesis and are reproduced here:

G.1 The use of recommendation information derived from non-reputation sources, and the subsequent usage of this information within a reputation system.

G.2 The integration of human and electronic entities, including the investigation of how these could be combined into one recommendation system.

G.3 The integration of identity and reputation, with a focus on how reputation information could be combined with identity information.

G.4 The use of contextual information within reputation systems. In particular, the ability to capture and make this information available for use in the evaluation of recommendations.

G.5 The ability to exchange recommendation information between systems.

1.4 Contributions

This thesis makes a number of contributions in the areas of reputation, reputation systems and access control. More specifically, this thesis:

1. Defines a terminology for reputation that describes reputation, trust and risk and then discusses the relationships between them. The term recommendation as it is used in the context of this thesis is also defined and discussed.

2. Defines a generalised reputation model that can be used to describe reputation in both online and offline contexts. This model introduces standardised terminology (based on published research) that is used throughout this thesis.

3. Defines an individual reputation context model that describes all of the contexts that an individual entity may possess. This model is useful in that it helps to describe the contextual nature of an individual and makes clear the need for multiple context support in reputation systems.

4. Provides a survey and taxonomy for reputation systems. The survey, conducted on both academic and commercial systems, was used to drive the development of the taxonomy. The taxonomy builds on five commonly accepted dimensions, while also providing nine new dimensions for those aspects of reputation systems that had not been covered widely previously, or never considered before. The taxonomy is subsequently used to build a classification of existing reputation systems, generating a large number of research leads.

5. Presents an architecture and prototype for a fully distributed recommendation system that supports both human and electronic service entities. The architecture and prototype encapsulate a paradigm shift in reputation systems and exhibit a number of unique ideas:

   (a) A novel and well-defined three layer architecture stack that underpins the design and implementation of GRAFT. The stack allows for a modular approach to building and integrating components into GRAFT, simplifying new design and development work. In particular, the separation of raw collection from integration into the peer-to-peer network allows for efficient nodes that only implement those aspects of the stack they require.

   (b) An exploration of identity and reputation integration. All recommendation information in GRAFT is tied to identity information. Knowing the identity of an entity is sufficient to be able to locate and consume their recommendation information. Previous systems have treated these two concepts as distinct, leading to classic two-step authenticate and authorise models. GRAFT instead considers all information about an entity when granting access, leading to better decisions.

(c) The integration of both human and electronic service entities into one reputation system. These two types of entities are treated identically within GRAFT. Their integration into one reputation system is possible due to the fact that identity and recommendation information have been combined using OpenID.

(d) The utilisation of a peer-to-peer network to store and replicate recommendation information. The peer-to-peer network distributes the load evenly across the peers that make up the network, whilst also ensuring robustness for profiles.

(e) The utilisation of both explicit and implicit reputation information. In particular, the use of information previously not regarded as being useful when combined with reputation such as demographic, social and derived information.

(f) The ability to retain the context in which reputation information was generated. Understanding the context of the information allows for it to be utilised in a meaningful way by a consumer.

(g) The utilisation of recommendation information in policy description and evaluation. The ability to combine recommendation information from multiple sources in the building of policies allows for flexibility that is otherwise not possible. For example, policies can build on the user's demographics and their professional standing, but can also utilise their social relationships.

6. Analysis of the performance of the GRAFT architecture. In particular, the performance is measured and significant factors affecting performance are identified using a series of experiments that utilise both the prototype and large-scale simulations. The simulations consider both the "perfect" state and increasing levels of churn and malicious peers.

1.5 Scope

This thesis has a dual focus. Firstly, the development of a standard terminology, models to describe reputation and reputation systems, and the development of a taxonomy for reputation systems. Secondly, the development, prototyping and evaluation of an architecture for the GRAFT distributed recommendation framework as enumerated by the research goals in Section 1.3. A number of supporting technologies are utilised to build the architecture, but are not the focus of this thesis. These technologies are introduced and further discussed in the appendix. In order to limit the scope of the potential work, the focus of this thesis is the development and evaluation of those components required to collect, store and make available recommendations.

1.6 Publications

Two full internationally peer reviewed conference papers, and one ERA A* ranked journal paper were published during my PhD candidature. These papers were used as a way to explore and peer review the ideas in this thesis. In particular, Chapter 2 is largely derived from the following paper that was published in an ERA A* ranked journal:

- **F. Hendrikx**, K. Bubendorfer, R. Chard, Reputation Systems: A survey and taxonomy. Journal of Parallel and Distributed Computing. 2014.

Chapters 3, 4 and 6 are in part derived from the following paper that was presented at an ERA A ranked conference:

- **F. Hendrikx**, K. Bubendorfer, Malleable access rights to establish and enable scientific collaboration, in: eScience (eScience), 2013 IEEE 9th International Conference on, IEEE, Beijing, China, 2013, pp. 334-341.

Similarly, Chapter 4 is also based in part on the following paper:

- **F. Hendrikx**, K. Bubendorfer, Policy derived access rights in the social cloud, in: eScience (eScience), 2013 IEEE 9th International Conference on, IEEE, Beijing, China, 2013, pp. 365-368.

1.7 Thesis Organisation

This thesis is organised as follows.

Chapter 2: Related Work discusses related work in reputation systems. In particular, this section introduces common terminology and provides a survey of existing academic and commercial reputation systems. A taxonomy is developed and then utilised to classify a number of reputation systems. This chapter concludes with an analysis of areas of research that are under-represented, and then considers existing systems which attempt to address some of these areas.

Chapter 3: Architecture develops a set of requirements based in part on the research goals. These requirements are then used to develop the GRAFT architecture. The design implications for keeping the profiles secure at each layer of the stack are also discussed. In particular, weaknesses and some areas for future work are identified here.

Chapter 4: Case Studies presents case studies examining how GRAFT could be utilised in different situations. In particular, this chapter introduces a number of case studies that are subsequently utilised in the implementation and evaluation chapters.

Chapter 5: Implementation introduces a prototype implementation of GRAFT, focusing on a subset of the case studies. The implementation includes the core components of GRAFT, and three of the case studies presented in Chapter 4.

Chapter 6: Experimental Results and Evaluation presents a series of experimental results. The experiments consider three key aspects of the architecture: recommendation sources, the utilisation of the architecture and the ability for the architecture to scale and efficiently provide profiles to consumers, even when under churn and with malicious peers.

Chapter 7: Conclusions concludes this thesis by reviewing the design requirements, contributions and areas for future research.

Chapter 2

Related Work

This chapter[1] discusses a standard terminology for reputation systems, a survey of existing reputation systems, the construction of a taxonomy and finally the use of the taxonomy to classify a number of existing systems. The classification is used to find gaps in the existing research and literature.

The requirement for trust and reputation is evident in many online systems. In online banking systems for example, the reputation of the service is implicit. In more open online business systems and electronic markets such as eBay [6], the explicit yet informal use of reputation through user feedback can be observed.

Building and maintaining a good reputation can be a significant motivation for contributing to online communities, be they scientific, business or socially oriented. It has been shown that a good reputation leads to more sales, at a higher value than might otherwise be possible [263]. Existing online reputation models, while diverse, are still in their infancy and are generally limited in scope, usually focusing on a single context for their information.

2.1 Terminology

2.1.1 Reputation and Trust

According to the Collins English Dictionary, reputation is *"the estimation in which a person or thing is generally held; opinion"*. Every person's opinion differs from every other person, making reputation a highly personal and subjective quantity [272]. Reputation is not what character someone has, but rather what character others think someone has. Mui et al. [220] define reputation as the *"the perception that an agent creates through past actions about its intentions and norms"*; this is the definition that will be used in this thesis.

[1]The work in this chapter is largely taken from the published paper F. Hendrikx, K. Bubendorfer, R. Chard, Reputation Systems: A survey and taxonomy. Journal of Parallel and Distributed Computing. 2014.

Reputation and trust (or trustworthiness) are commonly confused [222] and used as synonyms, even though their meanings are distinctly different. Jøsang et al. [159] define trust as *"the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible"*. The key concepts in this definition are dependence and reliability; these values are measured, in part, through an agent's reputation. It can therefore be said that trust can be established through the use of reputation. Arguably, a better reputation can lead to greater trust.

Risk is often undertaken in the hope of some gain or benefit. Risk can therefore be viewed as the situation where the outcome of a transaction is important to a party, however the probability of failure is non-zero [159]. Incorporating the previous notion of trust into this definition: the amount of risk that a party may be willing to tolerate is directly proportional to the amount of trust that the party has in the other party.

The main aim of reputation systems is therefore to support the establishment of trust between unfamiliar parties. Dellarocas [88] states that the aim of eBay's feedback mechanism, and in a generalised sense, all reputation systems, is to "generate sufficient trust among buyers to persuade them to assume the risk of transacting with complete strangers". Despotovic and Aberer [89] talk about "reducing the opportunism" and vulnerability of the two parties. Using a reputation system, a party may examine the history of another and decide that it will trust and interact with the other party. This decision is often called a "trust decision" [183].

Despite the best intentions of reputation systems, it is important to note that reputation only provides a "hint", and not clear evidence or proof of an agent's actual future intentions. Gaming the system, by generating a series of positive interactions to build up a good reputation, only to default at the last moment is not unheard of.

2.1.2 Recommendations

Recommendations are endorsements of the target agent by a third party. These endorsements are based on the target's characteristics, capabilities, affiliations, and reputation. In the past, letters of recommendation were used, for example, to introduce foreign envoys to a new court [315]. These letters were often written by a senior dignitary, and lent the envoy some of the dignitary's reputation. In a similar way, modern letters of recommendation are used to promote the suitability of an agent for a position or task.

Assuming the recommender is of good standing, a recommendation will help to establish trust between parties, as it might, for example, confirm that the agent is a member of a given organisation, or provide assurance that they are reliable. Conversely, if a recommender is not of good standing, a recommendation may carry less weight than if it came from a more reputable source.

As recommendations are based on information about an agent (for example, their reputation), they must be generated from prior knowledge, or past actions. Although the word recommendation tends to have a positive inclination, the intended usage throughout this thesis is such that any actions or information can be used to generate a recommendation. In this way, an agent can get positive, neutral and negative recommendations.

2.2 Reference Models

2.2.1 Reputation Systems

When discussing reputation systems, it is important to define the parties involved and their potential interactons. Figure 2.1 illustrates a generalised model of reputation systems that was designed to accommodate both real-world and online approaches to reputation.

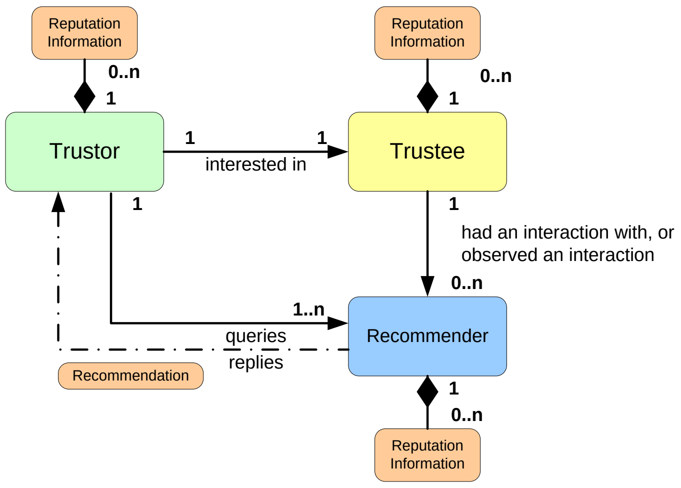

Figure 2.1: A Reference Model for Reputation Systems that accomodates both real-world and online approaches to reputation. The trustor is a party that wants to trust and interact with a target agent or entity, called the trustee. The trustor may then query $1..n$ recommenders that may have previously interacted, or observed an interaction, with the trustee.

The trustor is a party that wants to trust and interact with a target agent or entity, called the trustee [89, 183]. In order to make a trust decision about whether to trust the trustee, the trustor will need to evaluate the trustee's reputation [182]. It does this by first consulting its own internal reputation information, to see if it has previously interacted with the trustee and what the outcome was. However, if there was no previous interaction, the trustor will then query $1..n$ recommenders [274, 270, 183, 193] that may have previously interacted, or observed an interaction, with the trustee for their opinions.

A recommender may be an entity that provides information from its own history of transactions, or a system that either observed an interaction between two parties, or collects information from other sources [193]. A recommender with appropriate information may reply with a recommendation (sometimes also called feedback or a rating). A recommender may have a variety of first and third hand information; this is represented by the $1 : 0..n$ relationship between the recommender and its internal reputation information.

Using the reputation information obtained from the recommenders, the trustor is able to make its trust decision. The roles of trustor, trustee and recommender are completely interchangeable [183]; if the transaction proceeds, both parties will have their own reputation information that may subsequently be made available to other parties making similar decisions.

2.2.2 Reputation Context

Reputation is context dependent and relies on contextual information to give data meaning [24]. The definition of context with respect to reputation systems is often difficult to determine and there is no common definition used by researchers.

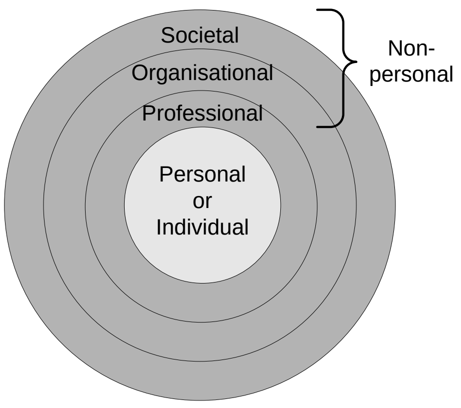

Figure 2.2: A Reference Model for Reputation Context. Starting with the innermost ring, reputation context can be personal, professional, organisational and societal.

Reputation systems are often discussed as utilising additional contextual dimensions [272], facets [130], or attributes [77] to provide greater meaning and usability to the information generated during a transaction. In order to unify this concept, the term contextual attributes has been adopted. Contextual attributes are like metadata, in that they help to describe the transaction in greater detail. For example, the date, the price, the buyer and the seller are all possible attributes of a transaction between two parties.

However, contextual attributes are not the entire picture. For that, a context is required, which is the domain in which the information was generated. Most reputation systems employ a single, or personal, context. In other words, most systems consider only the reputation of an entity in the "function" of the system (whether that be e-commerce, expert advice, or file sharing).

Reputation systems employing more than one context often add additional domains of information. For example, the addition of a social context to an existing personal reputation context can help to determine if an individual contributes to his or her community, and therefore if they are more trustworthy.

In an effort to summarise and clarify the relationship between context and reputation, a reference model based on a psychological framework of personal identity [299] has been developed. This reference model is presented in Figure 2.2. Starting with the innermost ring, reputation context can be personal (who), professional (what), organisational (which/membership) and societal (where).

Most online reputation systems focus only on the personal reputation of a person, whilst many real-world situations deal with non-personal aspects, such as a user's professional and organisational membership.

2.3 Reputation Systems Survey

In this section a number of academic and commercial reputation systems are examined. These systems were selected based on their impact or importance. Each reputation system has been considered in a systematic way so that comparisons could be drawn between them. The academic systems are organised in chronological order, while the commercial systems are organised alphabetically.

2.3.1 Academic Systems

The **Regret** [273, 272] reputation system is designed to operate within an electronic marketplace setting. The system utilises multiple contextual attributes and classifies information as coming from an individual, social, or an ontological dimension. The individual dimension considers information directly gathered from interactions between two entities. The information is fine grained and often relates to the frequency of overcharging, late delivery and quality of the transaction. The social dimension is an addition to the Regret system, where trust can be extracted from the groups and communities associated with the target entity. A key benefit of the social dimension is that it allows new and unproven entities to bootstrap their trustworthiness by belonging to reputable groups. Alternatively, because the entire group's reputation is associated with the behavior of its members, it is pertinent for a group's members to moderate the behavior of those associated with them.

The work also includes an ontological dimension, where reputation collected for atomic aspects are combined to construct more complex graph structures in order to derive further insight. Ratings, or impressions, are recorded as a value between positive and negative one. An entity's reputation is then the aggregation of the result of all transactions they have taken part in. When utilising the ontological dimension, each atomic aspect is calculated using individual and social dimensions, and then combined through a weighted graph for more complex evaluation. The computation of the ontological reputation, OR_{ij} is achieved through Eq. 2.1. Where each child in the graph is computed with a weight w_{xy} to establish a score. An example of this computation can be seen in Eq. 2.2 where the social dimension SR_{ij} for each aspect is weighted and used. The Regret system also employs a degree of reputation decay, called a forget factor, where only the most recent transactions are considered.

$$OR_{ij}(x) = \sum_{y \in children(x)} w_{xy} OR_{ij}(y) \tag{2.1}$$

$$\begin{aligned} OR_{ij}(good\_seller) = {} & 0.2 \times SR_{ij}(delivery\_date) \ + \\ & 0.2 \times SR_{ij}(product\_price) \ + \\ & 0.6 \times SR_{ij}(product\_quality) \end{aligned} \tag{2.2}$$

Confidant [52] incorporates a reputation system into a Dynamic Source Routing protocol in mobile Ad-hoc networks. The Confidant protocol is a structured system designed to identify and isolate misbehaving nodes in the network. The system allows entities to monitor the behavior of others, regarding their ability to manipulate information and correctly route, forward and participate in the protocol. Nodes are inherently trusted and malicious behavior is reported, resulting in a form of punishment.

An alarm message is generated when a node experiences, observes or receives a report of malicious behavior. Observed information is gathered by examining the interactions among neighbors. Alarm messages are then passed on to other nodes in the network to warn them of the misbehaving node. When an alarm message is received from another entity, their trustworthiness is considered before passing the alarm on. Ratings are stored as local lists and black lists at each node, and are potentially shared with friends. Confidant proposes to lessen the effect of false accusations with a reputation death property and revocation lists, meaning entities are capable of redeeming themselves over time as historic actions are removed from the system. No specific aggregation equations are published.

XRep [82] is an extension to GNUtella-like peer-to-peer (P2P) networks. XRep allows for the creation and maintenance of reputation for both resources and nodes in the network. Each node maintains a personal history for both resources and nodes. For resources, a simple binary rating is used, while for nodes a count of the number of successful and unsuccessful downloads is maintained. Reputation messages are piggybacked on existing connections and allow nodes to select resources based on criteria other than purely resource based. When deciding where to download a resource, a node first contacts its peers for their advice (using a poll operation), and then evaluates the responses. Once the node has selected the appropriate resource, it can evaluate potential nodes that offer the resource and select one based on reputation. Assigning reputation ratings to both resources and nodes in the network gives XRep a number of advantages. These include judging new resources by the nodes offering them, load balancing using a resource reputation rather than purely the node reputation, and whitewashing avoidance as changing pseudonyms removes nodes from being selected. A set of extensions to XRep, called X^{2}Rep [81] were created later and addressed some of the weaknesses in XRep.

EigenTrust [166] is a peer-to-peer (P2P) reputation framework that allows entities to decide which others they will trust when it comes to downloading files. It is fully decentralised, and utilises a distributed hash table overlay. Each entity maintains a personal history for other peers, which is simply the sum of positive and negative interactions they have experienced with them. These values are normalised between 0 and 1. An entity calculates the global Trust for another entity, T_{ij}, by using personal histories which are obtained from others in the network. These histories are weighed by the credibility of the reporting entity, as seen in Eq. 2.3, where e_{ix} denotes a local trust value of entity i for entity x. In essence, the system uses the direct experience of others and a local perception of the reporting peer to compute trust [144]. To compute trust in a distributed environment, the aggregation model represented by Eq. 2.4 is used. This is a component-wise method of computing the global trust of i which aggregates the trust each peer holds in i over a time period k. Where α is a constant less than 1 and p is used to add trust to new entities in the network.

$$T_{ij} = \sum_{k} e_{ix} e_{xj} \tag{2.3}$$

$$T_i^{(k+1)} = (1-\alpha)(e_{1i}T_1{}^{(k)} + \cdots + e_{ni}T_n{}^{(k)}) + \alpha p_i \tag{2.4}$$

P-Grid [24] is a peer-to-peer (P2P) platform for distributed information management. P-Grid is completely decentralised and self-organising. Information is spread across the environment among peers via a distributed search tree, similarly structured to distributed hash tables. As is the case with Confidant, entities in P-Grid are considered inherently trustworthy, and only malicious behavior is deemed relevant. Entities are able to forward complaints about transactions to others within the environment. These complaints are distributed in the form of messages to arbitrary entities. P-Grid implements a binary trust model, where entities are either trustworthy or not. When an entity wishes to evaluate the trustworthiness of a target, it performs a search for complaints. These are fed into a trust function such as that shown in Eq. 2.5. The function is used to determine whether i can trust the entity j. Where Cr denotes complaints received and Cf represents complaints filed. ${Cr_i}^{avg}$ and ${Cf_i}^{avg}$ are the aggregate of all observations the entity i has made over its lifetime. If the equation computes as true, then the entity is considered trustworthy, otherwise they are not.

$$Cr_x^{norm}(j)Cf_x^{norm}(j) \leq (\frac{1}{2} + \frac{4}{\sqrt{{Cr_i}^{avg}{Cf_i}^{avg}}})^2{Cr_i}^{avg}{Cf_i}^{avg} \tag{2.5}$$

PeerTrust uses a structured peer-to-peer (P2P) overlay network to host a distributed implementation of their transaction-based feedback system. The simulation used to demonstrate PeerTrust utilises P-Grid to distribute feedback scores. The system incorporates a combination of fundamental reputation sources, such as direct feedback, and the quantity of transactions performed, while weighting feedback with credibility. The work introduces two novel trust metrics, a community context factor and transaction context factor. The simulation presented in the work has entities generate a rating of either zero or one.

The trust of an entity i is computed by Eq. 2.6. Given N_i is the total number of transactions entity i has taken part in. P_{ij} denotes the other entity involved in the transaction. S_{ij} is the normalised level of satisfaction i received from peer P_{ij} from the transaction. Cr denotes the credibility of the feedback received from the entity P_{ij}. TF_{ij} represents the adaptive transaction context factor for entity i's jth transaction, and CF denotes the community context factor for i during a period of time. The normalised weighted factors α and β are the collective evaluation and the community context factors, respectively.

$$T_i = \alpha \sum_{j=1}^{N_i} S_{ij}Cr(P_{ij})TF_{ij} + \beta CF(i) \tag{2.6}$$

RateWeb [202] is designed to facilitate trust between Web services. RateWeb utilises a decentralised and unstructured approach. The system's goal is to provide a method in which Web services can reliably be used as independent components in a service-oriented enterprise without the intervention of humans.

When selecting a Web service to accomplish a task, the consuming entity queries the community for a list of suitable services. A set of eligible Web services are then returned to the consumer. The response also includes a list of past consumers that possess feedback for each service. Rather than acting as a centralised repository of feedback, the community acts as a directory of raters. Each entity stores a personal perception of each Web service it has invoked. The feedback is stored in a vector of values that represent the promised quality against the delivered quality of an attribute.

The reputation of a service s_i can be computed by an inquiring consumer through Eq.2.7. Where L denotes the set of consumers which have interacted with, and rated, the service s_i. $PerEval^x{}_i$ represents the personal perception a consuming entity has of the service s_i. The credibility Cr_x of each consuming entity, as viewed by the inquiring consumer, is within the interval $[0, 1]$. A reputation fader, or decay factor, D_f, is also incorporated and is a value within $[0, 1]$.

$$Reputation_{s_i} = \frac{\sum_{x=1}^{L}[PerEval^x{}_i D_f C_r(x)]}{\sum_{x=1}^{L} C_r(x)} \tag{2.7}$$

A credibility-based model is also included within the framework. In this model, each service contains a set of trusted entities to query when requesting ratings. If none of the group contain experience with the entity in question, each group member can refer the request to their own trusted set of entities.

R^{2}Trust [304] is a fully distributed reputation system for large-scale, decentralised overlay networks. The system is designed to incorporate reputation and risk to provide trust within an unstructured network. The reputation of an entity in the network is calculated by examining any direct interactions an entity may have had and obtaining recommendations from other peers. Recommendations are weighted using local trust values for the originating peers. The trust value assigned to any given peer is built using social relationships and considers the risk inherent in those relationships. This allows the framework to react quickly when the behavior of a given peer changes. R^{2}Trust determines quality of service as probabilistic ratings between zero and one. These values are then aggregated and accumulated to give an entity a reputation value. In order to filter out untrustworthy second-hand opinions, a credibility score is used to weight feedback during aggregation. A decayed trust value, DT_{ij} reduces the significance of feedback over periods of time, as shown in Eq. 2.8. Where $\lambda_k = p^{n-k}$ is the decay factor of time period k, and $0 < \lambda_k < \lambda_{k+1} \leq 1, 1 \leq k < n$. $e_{ij}^{t_k}$ denotes a local trust value of entity i for entity j over the time period t_k.

$$DT_{ij} = \frac{\sum_{k=1}^{n}(\lambda_k e_{ij}^{t_k})}{\sum_{k=1}^{n}\lambda_k} \tag{2.8}$$

2.3.2 Commercial Systems

Amazon [2] allows its registered users to write reviews on products. A user must first buy a product from Amazon, however they may then review any product carried by Amazon using a numeric rank (5 stars). Another Amazon user may then leave a boolean feedback rating of either “helpful” or “not helpful” for a product review. Reviews may be ordered by the number of “helpful” votes they have received. The reputation of a review author rises with each 'helpful' vote [193]. Amazon maintains a ranked list of reviewers based on their reputation, allowing them to apply badges such as “top 10 reviewer” and “top 500 reviewer” to their reviewers.

When considering online reputation systems, **eBay** is both well researched and much written about [261, 263, 262, 147, 213]. eBay is an online auction site, allowing sellers and buyers to trade goods through an auction process. At the end of a transaction, both parties to the exchange leave feedback for each other. This allows potential future parties to examine the reliability of any target party that has had a previous interaction. Feedback is left in the form of a single overall rating (Good, Neutral and Negative), a series of numerical ratings (for the following facets: Accuracy, Communication, Shipping Time and Shipping Charges) and a comment. The comment often provides further information about the actual quality of the item, shipping or any problems encountered.

Epinions [7] is a consumer products review site, founded in 1999. Users do not have to purchase anything and may write reviews on any product they chose, although they are encouraged to focus on new or previously unreviewed products. Good reviews can earn royalties on sales of the product that was the subject of the review.

Users do not have any visible reputation rating, however badges such as “top reviewer” and “popular author” are assigned to active users with good review ratings. Users are however able to maintain a list of other users that they trust. The number of users that trust a given user is publicly displayed, and acts as a form of reputation.

Slashdot [13] is a technology news website, founded in 1997. It is one of the earlier sites to utilise a reputation system. All registered users have an amount of “karma” that changes over time to reflect their level of activity, which includes the posting of articles and commenting. Users with a good level of karma are able to become comment moderators [249].

Stackoverflow is a dedicated Question and Answer site for developers, both “professional and enthusiast”. Users post questions that may then be answered by other users. Reputation points are awarded for all tasks (including asking and answering questions). However, more points are awarded for comprehensive answers as chosen by other users. As a user gains more points, they are able to access further features on the site, including the ability to vote up, vote down and act as a moderator (i.e. edit other user's questions and answers). A user that votes down a particular answer will lose 1 reputation point. Presumably this is intended to stop users from voting down too much.

Each user has a profile that features their reputation score, and how they achieved that score. The reputation score is represented using a discrete value; the more reputation points a user has, the higher their score.

Turkopticon [17] is an third-party reputation system for crowdsourced workers using Amazon's Mechanical Turk (AMT). It allows workers to check the reputation of work providers when viewing potential jobs. In particular, it allows a worker to view and rate a work provider on 4 facets of their behavior (Communicativity, Generosity, Fairness and Promptness).

Turkopticon is integrated into a worker's experience of AMT using a browser plugin. The plugin inserts the rating information into the AMT pages as they are rendered on the user's browser. Rating information is centrally maintained by Turkopticon, and maintained using worker input. Regular software updates improve the worker experience and resolve technical issues.

2.4 Reputation Systems Taxonomy

2.4.1 Related Work

In Mui et al. [222], the authors present a reputation typology. This typology includes only a small number of dimensions as it tries to combine reputational literature from a number of different disciplines.

A set of classification dimensions for trust and reputation models is introduced in Sabater and Sierra [274]. These are subsequently used to classify a number of well-known trust and reputation systems.

A taxonomy for peer-to-peer reputation systems is introduced in Marti and Garcia-Molina [207]. The goal of their paper is to organise existing ideas and work, so that design and implementation can be better achieved. They have 11 dimensions in three areas of interest: information gathering, scoring and ranking, and response.

A framework for the comparison of reputation-based trust systems for peer-to-peer applications is presented in Koutrouli and Tsalgatidou [182]. The authors investigate 14 dimensions, spread across three key areas of interest: information gathering, feedback aggregation and output. The focus of their paper is on peer-to-peer e-commerce, file sharing and cooperative applications.

In Wang and Vassileva [327], the authors introduce a classification of trust and reputation systems based on system structure. Their particular focus is on using trust and reputation information for web-service selection. The three classification criteria discussed in their paper are each directly related to the underlying architecture of the reputation system. Although aspects of system structure are identified by this work, other details are also considered in the classification.

A survey of trust and reputation systems is presented in Jøsang et al. [159]. The authors focus on and thoroughly investigate reputation calculation, and how it is implemented in currently deployed systems.

In Hoffman et al. [144], the authors present a survey of attack and defense techniques for reputation systems. An analytical framework for breaking down and comparing reputation systems is introduced in order to identify common issues. This framework considers aspects such as dissemination and calculation of reputation. They also discuss existing reputation systems in the context of security weaknesses and the defenses that are employed by these systems.

A taxonomy of attacks for peer-to-peer reputation systems is presented in Koutrouli and Tsalgatidou [183]. Their taxonomy breaks down reputation attacks into three primary categories: Unfair recommendations, Inconsistent behavior and Identity management attacks. The authors then present a series of defense mechanisms, and conclude with a roadmap for system designers.

Yao et al. [349] address common vulnerability issues in reputation systems. As part of their work they present a "decomposition" of reputation systems that examines dissemination and calculation in a number of systems.

2.4.2 Construction Methodology

The taxonomy was constructed using an iterative approach, as described by Nickerson et al. [225], where both an "empirical to deductive" and "deductive to empirical" approach are used. In this overall approach, the primary purpose of the taxonomy is used to drive the identification of the key dimensions and their characteristics.

Given that the purpose of the taxonomy was to contrast and compare the architecture, organisation and management of different reputation systems, the key dimensions focus on the structure, design and organisation of these systems. Dimensions in existing commercial and research reputation systems were examined, and those that related to the chosen dimensions were included. Further dimensions that are important as regards reputation systems architecture, organisation and management were then added.

The dimensions and characteristics were then used to classify a diverse set of reputation systems. The dimensions and characteristics were then further refined by ensuring that they could fully describe this set of systems. A number of other reputation systems were then classified. These were used to identify the dimensions and characteristics that required further work or were not defined clearly enough. This process was repeated until no further changes were required.

2.4.3 Taxonomy

The reputation taxonomy is given in Figure 2.3. The first level of the taxonomy distinguishes between explicit and implicit reputation systems. An implicit reputation mechanism represents systems that have not defined a reputation system, however reputation information is still employed by its members to assist in making decisions. The oldest and simplest form of implicit reputation system is the social word of mouth system as discussed by Dellarocas [88]. These "systems" have little or no structure, and have been used for centuries to ensure that participants in transactions remain honest, even when faced with the temptation to cheat the other party for short-term gains.

In more recent times, examples of implicit reputation systems can be found in social networks such as Facebook or LinkedIn [10]. Entities within a social network can extract some degree of trust for the information gathered through friends of friends. Although neither Facebook nor LinkedIn directly implement a reputation system, members of both systems are able to utilise reputable connections through friends within the environment. Another well-known implicit reputation system can be demonstrated in Google's [9] search engine. The order of the search results represents a ranking of pages, based on the reputation of each page. The reputation is determined by the number of links that point at the page, and where the links originate. A link originating at a page with a high reputation is likely to mean that the target page has some value. Pujol et al. [253], discuss utilising a similar kind of topology analysis in social networks to determine reputation.

Explicit reputation systems are those that have been purposely implemented to facilitate estimation of trust between members of an environment. They are typically used within an environment that relies on frequent interaction with a sufficiently sized, diverse set of members.

The second level of the taxonomy details the core dimensions of the taxonomy. The first five of these dimensions represent those aspects of reputation systems that are most often discussed in the existing literature:

Common Dimensions

1. **History**.

   A user's history is the set of stored information recording their past interactions and their outcomes. It is often used to determine the likely outcome of current, or future transactions, and is therefore central to the concept of reputation. A past transaction is often recorded in the form of an exchange between two entities, where each entity leaves feedback about the performance of the other in executing their duties and obligations. This feedback is often called a rating.

   - Personal: personal history is created and maintained using directly collected or observed information, leading to personal views of other entities. The subjective nature of these views means that others in the system may disagree with these views. This is sometimes also called localised or subjective history.

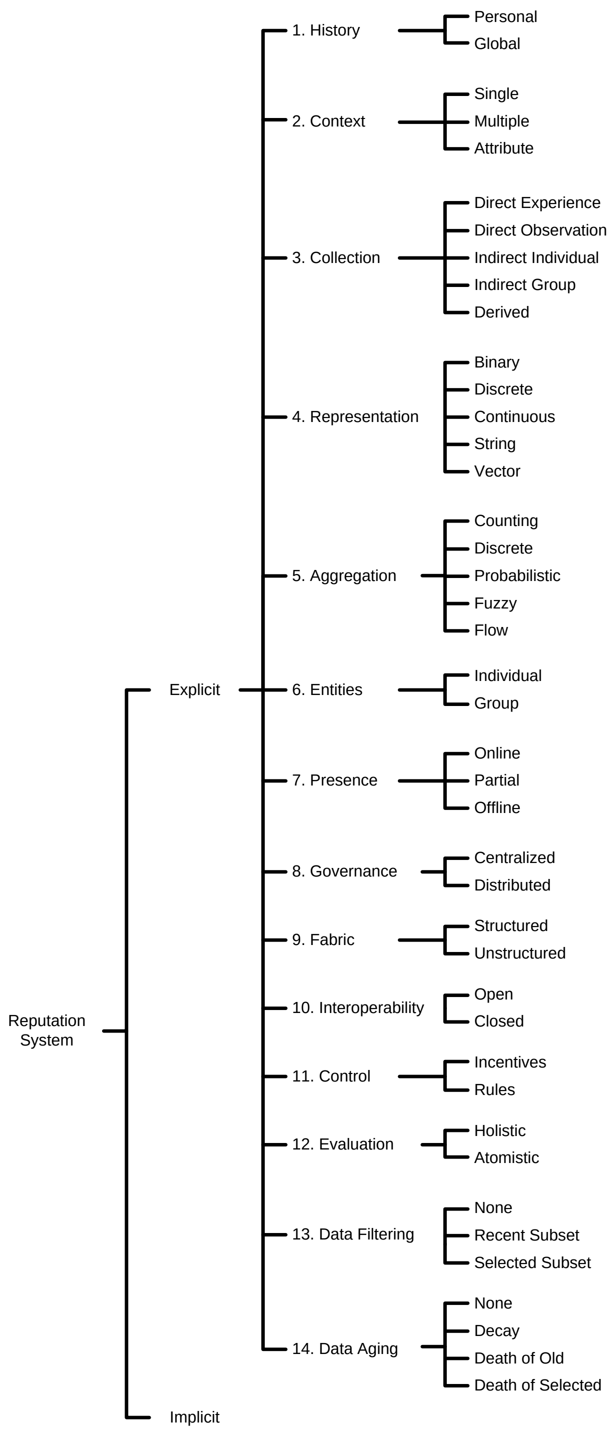

Figure 2.3: A Visual Representation of the Taxonomy.

- Global: global history is created and maintained from information shared by other members in a system, leading to a consistent, global view of every entity in the system.

Further discussion on these terms can be found in Mui et al. [222], Casare and Sichman [59], Sabater and Sierra [274], Koutrouli and Tsalgatidou [182], Wang and Vassileva [327], Marti and Garcia-Molina [207] and Zhao and Li [358].

2. **Context**.

Contextual information can give a lot of meaning to data by describing a range of details regarding how interactions take place [91]. In Section 2.2, it was proposed that context refer to the domain from which information is generated. In order to discuss and categorise reputation systems that utilise fine grained and transaction specific, contextual information, the term contextual attributes has been adopted.

Employing this definition, the majority of reputation systems can be classified as operating within a single context, as few systems employ information from distinct domains. However, contextual attributes are frequently used to give additional meaning to transactions and can greatly increase the usability of reputation information. Schlosser et al. [281] discuss contextual attributes when providing an example of goods being sold. They explain that not only the price and quality of an item are important when buying an item, but other information such as the delivery time and after sales services should also be considered.

In addition to a typical feedback score in which a peer's behavior in a network is analysed, Gupta et al. [130] include an explicit capability attribute when building a peer's reputation. Peers that provide desired resources to the network, such as, computing time, are given a greater supplementary reputation than those that provide few or no resources.

Reputation information can be generated from a vast number of transaction instances which are each accompanied by a significant amount of contextual information. Reputation systems have been categorised as either incorporating information from a single or multiple contexts as well as maintaining contextual attributes.

- Single: a single context is assumed or maintained within the system.
- Multiple: one or more contexts is maintained within the system. Support for multiple contexts is discussed by Bagheri and Ghorbani [32], Tavakolifard et al. [301] and Grinshpoun et al. [127].
- Attribute: Contextual attributes are maintained by the system. This property is sometimes called multi-faceted, dimensional, or attribute-based, and is further discussed by Gupta et al. [130], Sabater and Sierra [272], and Conner et al. [77].

Further general discussion on context can be found in Sabater and Sierra [274], Koutrouli and Tsalgatidou [182] and Wang and Vassileva [327].

3. **Collection**.

   For a reputation system to establish trust the behavioral information of entities needs to be captured. There are a number of techniques a reputation system can offer to enable the collection of information on interactions between entities.

   - Direct: information is generated explicitly either from an individual's personal interactions, or observation of other's transactions. This term is further discussed in Sabater and Sierra [274] and Mui et al. [222].
   - Indirect: information is obtained from other entities (either individuals or groups) based on transactions that the querying entity was not privy to. This is sometimes called witness information. This term is further discussed in Sabater and Sierra [274] and Mui et al. [222].
   - Derived: information is obtained from a source that was not explicitly designed to be used as a reputation source in the current context.

4. **Representation**.

   The format employed to describe, exchange and interpret reputation information. After investigating a number of frameworks and the method used to symbolise the information to members, the commonly used types of information are:

   - Binary: information is stored using boolean values. This term is discussed by Kinateder and Rothermel [177], Sabater and Sierra [274], Hoffman et al. [144].
   - Discrete: information is stored using discrete integer values. This term was also used by Hoffman et al. [144].
   - Continuous: information is stored as a floating point number. This term is discussed by Sabater and Sierra [274] and Hoffman et al. [144].
   - String: information is stored in textual form, allowing a wide range of data to be maintained. This term was also used by Kinateder and Rothermel [177] and Conner et al. [77].
   - Vector: information that is either provided by multiple sources or is explicitly separated for individual use. This term is discussed by Koutrouli and Tsalgatidou [182].

5. **Aggregation**.

   Aggregation describes how a reputation score for an entity is computed. The simplest form of reputation aggregation is the summation of all of the positive and negative ratings for an entity [159]. Each positive rating adds one to the sum, while each negative rating subtracts one. The final rating can be used to rank all the entities in a system. A slightly improved approach is to average all of the ratings to produce a single rating for each entity. Averaging is often used in conjunction with normalisation to evaluate entities on a specific scale. Weighting the ratings by factors such as, age, reputation of the source or importance of a transaction, can provide further ways to enhance this approach. Summation, averaging, weighting and normalisation are common aggregation methods and fall into a single class of simple computation called counting.

A different approach is to consider reputation as multiple discrete values as opposed to continuous values. Abdul-Rahman and Hailes [21] present a model where an entity is judged to be either "Very Trustworthy, Trustworthy, Untrustworthy and Very Untrustworthy". This is simpler for humans to work with [159], but is not optimal during computation, as the discrete rating values must be converted using look-up tables, weighted and converted back to discrete values.

Another class of aggregation involves fitting prior knowledge about another entity into a probability model and computing the likelihood of a hypothesis being correct. The hypothesis often takes the form of "is entity x trustworthy?". In other words, knowledge of prior events is used to predict future outcomes.

Aggregation using fuzzy logic is discussed by Song et al. [292]. In their system, fuzzy rules are used to determine the reputation score for both buyers and sellers.

- Counting: reputation is computed by either summing positive and negative ratings, or averaging ratings. The ratings may be weighted to provide a bias towards, for example, recent ratings or those from more reputable sources. This term was also used by Yao et al. [349].
- Discrete: reputation is computed by converting discrete rating values using look-up tables. This term is also present in Jøsang et al. [159].
- Probabilistic: ratings are fitted into a probability model and used to predict the likelihood of a hypothesis being correct. This term is discussed by Ruohomaa et al. [270], Koutrouli and Tsalgatidou [182], Hoffman et al. [144] and Yao et al. [349].
- Fuzzy: fuzzy logic is used to process or compute ratings, allowing these systems to work with a degree of uncertainty. Jøsang et al. [159] and Yao et al. [349] use this term.
- Flow: reputation is computed by examining the flow of transitive trust. This term was also used by Jøsang et al. [159] and Yao et al. [349].

Good general discussions on aggregation can be found in Jøsang et al. [159], Hoffman et al. [144] and Yao et al. [349]. Discussions on preserving privacy during reputation aggregation can be found in Pavlov et al. [243], Steinbrecher [294] and Gudes et al. [128].

Uncommon Dimensions

In addition to the five common dimensions already presented, further aspects of reputation systems that are not widely discussed in the existing literature have been identified. Each of these aspects were captured as dimensions in the taxonomy and each has been investigated and discussed below:

6. **Entities**.

   Entities are the primary focus, or target of a reputation system. The targets of a reputation system are typically either people or resources (for example, books or films) [327]. However, with the expansion of reputation systems there are now websites that need to cater to both. For example, Amazon allows members to rate both reviewers and resources, and Damiani et al. [82] talk about combining peer and resource reputations in peer-to-peer networks. Both people and resources are typical first class members of a reputation system. They have some similar reputational requirements, and are therefore not considered distinguishing factors in this taxonomy. Mui et al. [222] presents a reputation typology that includes the notion of "individual and group reputation". Reputation can be collected and accrued for these two different types of entities. The entities category has been included to provide a basis to differentiate between systems that operate on individuals and those that function over groups of entities.

   - Individual: these systems are focused on people or specific resources. This concept is further discussed in Wang and Vassileva [327].
   - Group: these systems are focused on groups rather than individuals. Groups can be both formal and informal in nature, with the former assuming some of the characteristics of an organisation. Systems that utilise groups are discussed by Mui et al. [222], Gal-Oz et al. [116] and Tong and Zhang [306].

7. **Presence**.

   Presence describes how closely a reputation is tied to its underlying reputation system. In most early reputation systems, an entity's reputation information was only available to be fetched or updated by a central server. Later systems distributed the reputation information, however the entity holding the information is still required to be online.

- Online: those systems that require the continuous presence of authority in order to be able to distribute reputation information. This is the default position for most reputation systems.
- Partial: those systems that do not require the continuous presence of authority in order to be able to distribute reputation information. Initial discussions can be found in Ismail et al. [154] and Prashant and Dasgupta [90].
- Offline: those systems that do not require the presence of authority in order to be able to distribute reputation information. This is a logical extension to the other categories in this dimension. However, it should be noted that there are not thought to be any systems that could be classified as offline.

8. **Governance**.

   Reputation systems are volatile environments with entities and information changing frequently. In order for the system to function properly, providing trust within a community, some level of authority is required. Governance describes that authority, and in particular, how the system is controlled.

   - Centralised: a centralised group or organisation manages the system. Most commercial reputation systems exhibit centralised governance, including Amazon, eBay and ePinions. In each instance, the underlying architecture may well be distributed, but the management is most likely by a single organisation.
   - Distributed: multiple entities working together, often with no centralised management. Entities within such a system may come and go as they please. Most recent peer-to-peer systems display distributed governance.

9. **Fabric**.

   Fabric describes how the nodes of the reputation system are organised. The organisation of a reputation system is a fundamental attribute, allowing systems to be easily categorised and differentiated between.

   - Structured: new nodes are assigned a location and a set of neighbors in an organised fashion when connecting to a network [207]. The topology may be formed using an overlay and therefore unrelated to the underlying network [284].
   - Unstructured: networks do not exhibit any organised arrangement and generally allow new nodes to connect randomly [207].

10. **Interoperability**.

    Interoperability describes the underlying principles by which the system operates and shares information. At present, most commercial reputation systems are tightly controlled and the information contained within them is not shared with third parties. This is because they consider their reputation information to be commercial property. As a result, members with good reputations are typically reluctant to leave and build up a new reputation with another provider.

    - Open: entities may freely access and utilise the reputation information contained within a system using data standards or APIs. Most academic systems fall into this category.
    - Closed: reputation information is proprietary and not usually shared outside of a system. Most commercial systems would fall into this category.

11. **Control**.

    Control describes the manner in which a reputation system motivates and controls entities to act in a desired manner, and is a fundamental aspect of any implementation. Arguably reputation systems are themselves are socially corrective, however this dimension is only concerned with explicit rules and incentives/disincentives used within a reputation system in order to get entities to behave in the desired manner.

    - Rules: an entity is forced or limited to act only within a prescribed manner.
    - Incentives/Disincentives: an entity is motivated or guided using rewards and punishments to obtain appropriate behaviors. Incentives are further discussed in Jurca and Faltings [163], Wongrujira and Seneviratne [331] and Marti and Garcia-Molina [207].

12. **Evaluation**.

    When obtaining or viewing the available reputation information for a given target entity (the trustee), reputation systems may provide two different views of previous transactions.

    - Atomistic: a detailed transaction-based view that potentially shows all of the interactions that the trustee has had. This can be used to window-in on only one aspect of the available reputation information.
    - Holistic: all of an entity's interactions are considered and weighed to provide a single, overall view of the trustee to the trustor. No detailed information is provided, only the summarised information is available for evaluation.

13. **Data Filtering**.

    - None: data provided by the system is not limited or filtered in any way. Trustors may utilise the full available history for a given trustee.
    - Subset: data provided by the system is limited by the application of data filtering. Trustors may utilise only a subset of all available history for a given trustee. A subset is most usually based on the age of the data, or some manual selection.

14. **Data Aging**

    Data Aging essentially reduces the confidence of information as time passes and more information is collected. The decay in value of information allows entities to distance themselves from historic behavior. Information decay helps prevent attacks on the reputation system in which entities build a sufficient level of trust and then begin acting maliciously. As the most recently gathered information is given the largest weight of confidence, new negative behavior will have the largest impact when making decisions.

    - None: reputation information is retained indefinitely.
    - Decay: reduces the confidence and granularity of older reputation information as time passes. Koutrouli and Tsalgatidou [182] discuss this idea further.
    - Death: an extension of decay that allows older reputation information to be discarded [351]. Information is usually discarded based on age, or a manual selection.

2.4.4 Classification of Reputations Systems

In this section the reputation taxonomy is used to classify a large number of academic and commercial reputation systems, see Tables 2.1 and 2.2. In these tables a '?' in a cell indicates that information on the characteristic was not available in the published work. Multiple entries in a cell indicate that multiple characteristics are supported.

1. **History**. Only a small number of the reputation systems use personal history, while the majority utilise global history. As the name implies, personal history is formed from the personal experiences of a single entity, and is the only type of history that can be fully relied upon. Lai et al. [187] argue that personal history does not scale well, as the chance of interacting repeatedly with the same entity is fairly small. As a result, personal history is less efficient, as other entities are not learning from your experiences. Jurca and Faltings [163] argue that personal history can be used

Reputation System	01. History P=Personal G=Global	02. Context S=Single M=Multiple A=Attribute	03. Collection DE=Direct Experience DO=Direct Observation II=Indirect Individual IG=Indirect Group DN=Derived	04. Representation B=Binary, D=Discrete C=Continuous S=String, V=Vector	05. Aggregation C=Counting, D=Discrete P=Probabilistic Z=Fuzzy, F=Flow	06. Entities I=Individual G=Group	07. Presence O=Online P=Partial X=Offline	08. Governance C=Centralised D=Distributed	09. Fabric S=Structured U=Unstructured	10. Interoperability O=Open C=Closed	11. Control I=Incentives/Disincentives R=Rules	12. Evaluation H=Holistic A=Atomistic	13. Data Filtering N=None R=Recent Subset S=Selected Subset	14. Data Aging N=None, C=Decay D=Death of Selected T=Death of Old
Histos [351]	P	S	DE&II	C	C	I	O	?	?	O	?	H	R	N
Sporas [351]	G	S	II	C	C	I	O	?	?	O	I	H	N	T
Regret [273, 272]	G	M‖A	DE&II&IG	D&V	C	I	O	?	U	O	?	H	N	C
P-Grid [24]	G	S	DE&II	B	C	I	O	D	S	O	I	H&A	N	N
Beta [158]	?	S	?	?	P	?	O	C	U	O	?	?	?	?
Confidant [52]	G	S	DE&DO&II	?	?	I	O	D	S	O	I	H	N	T
XRep [82]	P	S	II	B&D‖C	C	I	O	D	U	O	I	H	N	N
EigenTrust [166]	G	S	DE&II	C	F	I	O	D	S	O	I	H	S	N
Gupta et al. [130]	G	S‖A	DE&II	D&C	C	I	O	C	U	O	I	H	N	C
TrustMe [289]	G	S	II	?	C	I	O	D	U	O	?	H	N	N
PeerTrust [335]	G	M‖A	DE&II	C	C	I	O	D	S	O	I	H	R	N
Ismail et al. [154]	G	S	DE	D	na	I	P	C	U	O	na	H	S	N
Pride [90]	P	S	DE	?	?	I	P	D	U	O	I	H	N	N
TrustGuard [293]	G	S	II	C	C	I	O	D	S	O	I	H	N	T
FuzzyTrust [292]	G	S	DE&II	?	Z	I	O	D	S	O	?	?	N	C
Travos [242]	?	S	?	?	P	?	O	D	U	O	?	?	?	?
PowerTrust [364]	G	S	DE	B&C&V	F	I	O	D	S	O	I	H	S	N
Gal-Oz et al. [116]	G	na	DE&II	V	C	I&G	O	?	?	O	I	H	?	C
Coner et al. [77]	G	S‖A	DE	C&S	C	I	O	D	U	O	I	H&A	N	N
H-Trust [358]	P&G	S	II	B‖C	C	I&G	O	D	S	O	?	H	S	C
RateWeb [202]	G	S	DE&II	D&C&V	C	I	O	?	U	O	I	H	R	N
Tong and Zhang [306]	G	na	DE&DO	C	C	I&G	O	?	U	O	I	na	N	N
R^2Trust [304]	P&G	S	DE&II	C	P	I	O	D	U	O	I	?	?	C
ReDS [29]	G	S	DE	C	C	I	O	D	S	O	?	?	N	N
Tulungan [239]	G	S	II	?	?	I	O	C	?	O	I	?	?	?
PerContRep [339]	G	S	DE	?	C	I	O	C	?	O	?	?	N	C

Table 2.1: Summary of Academic Reputation Systems

Reputation System	**01. History** P=Personal G=Global	**02. Context** S=Single M=Multiple A=Attribute	**03. Collection** DE=Direct Experience DO=Direct Observation II=Indirect Individual IG=Indirect Group DN=Derived	**04. Representation** B=Binary, D=Discrete C=Continuous S=String, V=Vector	**05. Aggregation** C=Counting, D=Discrete P=Probabilistic Z=Fuzzy, F=Flow	**06. Entities** I=Individual G=Group	**07. Presence** O=Online P=Partial X=Offline	**08. Governance** C=Centralised D=Distributed	**09. Fabric** S=Structured U=Unstructured	**10. Interoperability** O=Open C=Closed	**11. Control** I=Incentives/Disincentives R=Rules	**12. Evaluation** H=Holistic A=Atomistic	**13. Data Filtering** N=None R=Recent Subset S=Selected Subset	**14. Data Aging** N=None, C=Decay D=Death of Selected T=Death of Old
Advogato [1]	G	S	DE	D	C	I	O	C	?	C	I	H	N	N
Amazon [2]	G	S	DE	B&D&S	C	I	O	C	?	C	I	H	S	N
Couchsurfing [3]	G	S	DE	D&S	C	I	O	C	?	C	I	H&A	S	N
Digg [4]	G	S	DE&II	D	C	I	O	C	?	O	I	H	N	N
eBay [6]	G	S‖A	DE	D&S	C	I	O	C	?	C	I	H&A	S	T
ePinions [7]	G	S	DE	D&S	C	I	O	C	?	C	I	H&A	S	N
MTurk [11]	G	S‖A	DE	D	C	I	O	C	?	C	I	H&A	N	N
PageRank (Google) [9]	G	S	DE	D&C	F	I	O	C	?	C	na	H	N	N
Reddit [12]	G	S	DE	D	C	I	O	C	?	C	I	H	N	N
SlashDot (Karma) [13]	G	S	DE	D&C	C	I	O	C	?	C	I	H	N	N
Stackoverflow [14]	G	S	DE&II	D	C	I	O	C	?	C	I	H&A	N	N
TrustedSource [15]	G	S	DE	D	C	I	O	C	?	C	R	H&A	N	C
Trustribe [16]	G	S‖A	DE&II	D&S	C	I	O	C	?	O	I	H	N	N
Turkopticon [17]	G	S‖A	DE	C&S	C	I	O	C	?	C	I	H&A	N	N
Yelp [19]	G	S	DE	D&S	C	I	O	C	?	C	I	H&A	S	C

Table 2.2: Summary of Commercial Reputation Systems

as a form of competitive advantage in certain circumstances. They suggest that a payment scheme can be used to incentivise truthful sharing of information with others.

RateWeb [202] proposes model where history can be fetched from a "rating clique". However, the members of these groups still act as individuals.

Global history is available to either everyone, or the members of a selected group. Lai et al. [187] notes that although global history scales well, it is vulnerable to some types of malicious attacks. Over time, a global history should give a consistent, long-term view of an entity that approximates a relationship [261].

2. **Context**. The majority of reputation systems only employ information from a single context. The systems that allow information from multiple contexts to be utilised by members differ substantially.

   PeerTrust [335] incorporates two additional forms of contextual information; transactional attributes and a community-based context. The transactional attributes include the value of the trades being participated in, such that users can establish which trades are most relevant to their current situation. The community-based context is used to measure the level of participation within the community, for example, whether an entity often provides feedback.

   Regret [273] expands on PeerTrust and stores reputation information in the form of a vector. Individual reputation values are associated with each contextual attribute, such as the chance to overcharge, deliver late or provide a low quality item. Sabater and Sierra have also extended the Regret system to include a social context for reputation information, where trust is extracted from groups and communities (the professional and organisational rings from the context model) to which an entity belongs [272].

3. **Collection**. The majority of systems use some combination of direct experience and individual indirect. Tong and Zhang [306] argue data is more reliable if collected directly rather than through a third party. The authors state that inaccuracies and lying reputation sources are reason enough to promote the direct observation collection technique. Indirect approaches involve information being obtained from other entities based on transactions that the querying entity was not privy to.

   EigenTrust [166] explains that utilising both an individual's personal experience as well as other's indirect experience allows for making better decisions. Third party entities can be discovered and queried in a variety of ways. However, information received from others that have been discovered through either single or multiple-hop transitive trust chains should be more reliable than that discovered by querying a random entity in the network [343].

RateWeb [202] proposes a collection model that involves a set of trusted entities being contained within each entity. When making a decision on whether or not to interact with another entity in the environment, the trusted entities are consulted for their historic actions with the target entity. RateWeb also suggests another method of indirect collection using groups and communities.

Mui et al. [222] talk about "prior-derived" reputation; however this is reputation based on prior beliefs and prejudices. As expected, none of the systems surveyed utilised derived data. Deriving information from an open repository can provide a unique view of an entity's disposition in that context. An example of this could be using an online literature library such as CiteSeer [121], to determine how prominent an entity is in the computer science domain.

4. **Representation**. Commercial systems mostly favor a representation of the reputation that utilises both a numerical value and textual content. Academic systems however tend to exhibit a range of representations, although textual content is not as prevalent as in the commercial systems.

5. **Aggregation**. Most commercial, and many early academic systems use a simple counting-based aggregation, either by summing all of the ratings together, or providing an average. None of the systems surveyed utilised a discrete model of aggregation, probably because of the non-optimal computation aspects of aggregation with this model.

   Beta [158] and Travos [242] both implement a probabilistic approach using the Beta probability density function, as this is considered suitable for processing binary ratings [349].

   EigenTrust [166] utilises a flow approach to calculating global reputation values. In particular, global reputation values are calculated using the "left principal eigenvector of a matrix of normalised local trust values". The local, or personal, trust values in EigenTrust are sums of the positive and negative ratings. PowerTrust [364] employs a similar approach to EigenTrust, but uses a Bayesian method to calculate the personal trust values.

   A fuzzy approach to aggregation is utilised by FuzzyTrust [292] when aggregating personal trust ratings.

6. **Entities**. The concept of group reputation is introduced by Mui et al. [222]. Barring three exceptions, all reputation systems that were classified in the taxonomy focus on individuals. These individuals could be either people or specific resources.

The first exception is the work presented by Gal-Oz et al. [116], where communities are broken down into smaller sub-communities the authors call "knots". Knots are formed from community members who have strong trust relationships amongst themselves. The reputation information within a knot is therefore more valuable and is given a higher weighting.

The second exception is the work presented by Tong and Zhang [306]. In their paper, they propose using direct observation of the size of the group to determine its reputation. If individual entities are seen to be joining a group, then clearly that group has a positive reputation, and vice-versa. Although not explicitly stated, they are in fact looking at the size of the group over a period of time. A simplified view of group reputation is basically the change in the number of members over time.

The last exception is discussed in Zhao and Li [358], where nodes are able to calculate the reputation of other entities, either individuals or groups, using their own private history and choice of algorithm.

Finally, although Regret does not support group entities directly, Sabater and Sierra introduce the idea of a "neighborhood reputation" that is based on the reputation and relationships of neighboring entities to the target entity. A given entity therefore inherits a reputation by default, allowing even entities that are less well known to have reputation.

7. **Presence**. Of the systems examined, none is fully offline, and only Ismail et al. [154] and Pride [90] support a partial presence. The former talks about distributing reputation information to third-parties via certificates. An authority is utilised to create the certificates, but not needed to distribute and interpret the information contained within them.

   The latter introduces the Pride reputation system which has been constructed for decentralised peer-to-peer networks. The system enables self-certification of entities with digital certificates and employs an elicitation storage protocol to distribute reputation information.

8. **Governance**. Most commercial reputation systems exhibit centralised governance, including Amazon, eBay and ePinions. The terms *centralised* and *distributed* are most often used in conjunction with the description of reputation system architecture such as in Wang and Vassileva [327], Jøsang et al. [159], Gupta et al. [130], Dutta et al. [101] and Wang and Li [321]. However, it is often difficult to establish with any degree of certainty how a given reputation system is actually implemented, particularly commercial systems that rarely provide operational details. For example, in Wang and Vassileva [327] eBay is noted as being centralised, when in reality it may

well be distributed in order to cope with the traffic load. However, it is possible to say that the system has centralised governance.

9. **Fabric**. There is a good mix of systems showing both structured and unstructured characteristics. PowerTrust [364] employs a structured approach to reputation collection and aggregation. The system utilises a trust overlay network to model transactions and nominates trustworthy entities as power nodes, responsible for aggregating global reputation scores.

10. **Interoperability**. Although reputation information within academic systems is often freely accessible, none of the systems surveyed had any explicit support for importing or exporting reputation information.

    For a short period of time, Amazon allowed its members to import their feedback scores from eBay, effectively removing the need to establish a new reputation. However, once legal action was taken, Amazon was forced to remove this functionality [261].

11. **Control**. The control category is often overlooked due to an underlying assumption that trustworthy users are rewarded. The practice of disincentivising entities in a reputation system is also not trivial. The following examples demonstrate a disincentive technique to promote good behavior in the environment, however the environment itself is typically restricted. Confidant [52], for example, utilises the disincentive principle, by placing consequences on badly-behaved routers. Once entities are discovered that appear to be acting maliciously, Confidant works to restrict the use of that router in future transmissions.

    P-Grid [200], is another system that punishes rather than rewards certain behavior. Entities within the system are assumed to be co-operative, meaning that only malicious behavior has an effect on one's reputation score. Both of these examples occur in a system where entities can easily be neglected and avoided. Other environments that use reputation systems may not be able to enforce such penalties as easily. For example, in a Grid setting, the process of excluding an entity from joining a virtual organisation can be extremely costly if they provide a sufficiently limited resource.

12. **Evaluation**. All systems surveyed for this taxonomy implement a holistic evaluation of the trustee. A small number of the systems also support an atomistic evaluation method, allowing the user to drill-down on particular aspects of the available information.

13. **Data Filtering**. Filtering is employed by a few systems. RateWeb [202] uses an approach where the trustors in the system apply limitations on data to determine

what is too old to be considered useful. RateWeb includes a method called "reputation fading". Each rating is time-stamped, allowing newer feedback to be given a higher weighting when computing reputation scores. This is considered data filtering and not data aging because the data is not discarded, but rather filtered by the trustor.

14. **Data Aging**. A number of systems implement Data Aging. The Regret [272] system provides a time-dependent method to calculate an individual reputation. As information gets older, its weight in the calculation diminishes. The authors cite Karlins and Abelson [170] as support for the feature. Zacharia et al. [351] implements a method to age and remove information through a "dumping function". The authors state that larger amounts of feedback increase the accuracy of the reputation system. Due to entities being able to alter their behavior at will, the authors assert that it is beneficial to disregard old ratings to move the behavior predicted by the reputation system closer to an entity's current performance.

2.4.5 Analysis

The application of the taxonomy to a number of academic and commercial systems in the previous section has identified several areas of reputation systems that are currently under-represented in research:

A.1 Contextual information is still largely under-utilised within reputation systems. Reputation information is context dependent, however few reputation systems support more than a single context. Wider investigation is required into the maintenance and utilisation of information from multiple contexts. In particular, the aggregation of contextual rating information into a single value for consumption, or maintaining distinct values. The importance and procedures behind the exchange of reputation information between distinct reputation systems is a weak area of the current research.

A.2 Derived information sources require wider investigation. In particular, the identification of derivable reputation sources, the ability to aggregate and utilise the derived information within a reputation system, and the policies to integrate and embed distinct sources. The value of including derived information from abstract sources that were not explicitly designed to generate information for decision making is a rich area for new research.

A.3 Individual entities are pervasive in current reputation systems. There is little material on group entities. Group entities have an important role to play in future reputation systems. In a virtual organisation, individual and group entities may join together temporarily to solve a common problem or work on a task. Reputation can play a key role in such environments.

A.4 Although partial presence is exhibited by two of the systems that were examined, none of the systems had the ability to distribute reputation information fully-offline. The ability to operate fully-offline would allow for robust decentralised reputation systems.

Although not explicitly addressed by either the survey or the taxonomy, the following gaps were also raised during the analysis of existing reputation systems:

B.1 The integration of human and electronic entities within reputation systems was not discussed in the taxonomy, however it is an area that requires wider research. In particular, how human and machine users could be incorporated into a single reputation system, and any complexities associated with such a merger.

B.2 Implicit support for the import and export of reputation information is another area for further research. The ability to import and export reputation information becomes more complex as the contexts vary. For the base-case of exchanging information between systems with similar contexts, we only require a set of standards that describe how reputation information should be encoded and exchanged. For more complex cases, where systems have differing contexts, this would require a set of standards that enumerate the contexts, and a way to generalise reputation values as they move between these contexts.

B.3 The integration of identity and reputation is not widely considered, and requires further thought. In particular, storing reputation information along with identity information would help in the transfer of reputation between different contexts, and would aid in the bootstrapping of existing users on a new system. Further, the ability to centralise both identity and reputation information would help to break down reputation silos and make reputation information more useful in a wider sense.

B.4 The ability to utilise reputation as a service is only just in its infancy. As with any other service, being able to query and update reputation as a service would be useful in many environments.

B.5 The ability for a reputation system to support not only reputation, but also recommendations. In particular, the ability to use personal, professional, organisational and societal contexts would aid in the utilisation of the information by different consumers.

Research Goals

The research goals for this thesis reframe a number of gaps that have not been adequately addressed in the existing research and literature:

G.1 The use of recommendation information derived from non-reputation sources, and the subsequent usage of this information within a reputation system.

G.2 The integration of human and electronic entities, including the investigation of how these could be combined into one recommendation system.

G.3 The integration of identity and reputation, with a focus on how reputation information could be combined with identity information.

G.4 The use of contextual information within reputation systems. In particular, the ability to capture and make this information available for use in the evaluation of recommendations.

G.5 The ability to exchange recommendation information between systems.

2.5 Related Systems

This section discusses and examines existing reputation systems that attempt to address the same issues as the research goals of this thesis. These reputation systems are discussed in chronological order, starting with the oldest system first.

In [329, 330], Windley et al. introduce Pythia, a reputation-based authorisation system. Reputation information is obtained from a number of source applications and stored in a central repository. Relying Parties (RPs) are then able to query the system and obtain reputation information about users that has been processed through a rules engine.

The approach presented in Pythia is limited. In particular, the architecture is centralised, and exhibits strong centralised governance. Relying Parties are presented with a pre-calculated reputation, limiting the flexibility of the information usage. For example, knowing that an individual possesses a grade better than 90% of the population is useful in nationwide comparisons, but does not help to determine their individual level of attainment. It also assumes that reputation values from different applications can be normalised consistently. While this approach makes sense in a small system with a limited number of applications, it does not scale because it is harder to reconcile reputation values across increasing numbers of systems. Finally, Pythia is focussed only on reputation-based models, and does not consider other recommendation information such as a user's demographics or their social connections.

Pingel and Steinbrecher [245] introduce a centralised system that collects and makes available reputation information. However, in this instance the information is collected from, and distributed to, participating communities. The authors call this "Cross-Community Reputation", and argue that reputation information from multiple communities, and hence sources, is more accurate. Users register a single pseudonym, however in order to preserve privacy, access to another community is via a credential that only the identity provider is able to map back to the original pseudonym. While this preserves privacy, it depends on a single centralised identity provider and reputation manager. This work is limited by its centralised governance and an underlying assumption that reputation values from different sources can be normalised. This work also separates identity and reputation, by storing reputation values in a stand-alone reputation manager.

In [127], Grinshpoun et al. introduce a Cross-Community Reputation (CCR) model. They also argue that reputation information sourced from multiple communities is more accurate, and removes the need to bootstrap a new reputation for every new community. The model discusses pre-conditions for sharing, reputation conversion and mapping of information from one community to another. Reputation information can be shared amongst different communities. A separate policy controls how much information is shared between the communities.

This work is limited in that it only talks about a model for reputation sharing and does not provide an architecture or discuss implementation experiences. Reputation is discussed as being an important component of identity, however the model does not provide any further guidance on these concepts. OpenID is discussed briefly, but not utilised in any way. The authors note that their model does not deal with time-dependencies such as decay of older reputation data.

Augo et al. [26] introduce a federated reputation model that focuses on the user trust problem: once a user is authenticated, how can trust be established between two parties on the same Relying Party (RP). The approach presented in the paper discusses building a federated solution that builds upon a single identity provider within a federation of services. Reputation is held by a separate reputation manager within the identity provider. The services provide a snapshot of the user's reputation to the reputation manager as the last step in the authentication sequence. Reputation is considered a four-dimensional vector ("Ego", "Reward", "Fear", "Profit"), and as such values can be compared across different contexts by considering the different dimensions.

Once again, this work is limited in that there is no actual architecture or implementation. The federated model works well across a small set of well-defined services, but does not scale. The separation of the identity provider and the reputation manager fails to recognise that the models should be combined, while the construction of four-dimensional vectors is subjective, requiring careful human intervention during construction. Finally, there is an underlying assumption that reputation values from all contexts can be normalised into one consistent representation.

In [205, 307], Marmol et al. discuss enhancements to OpenID to support the collection of reputation information. The system works by displaying reputation information about a Relying Party (the service provider) when a user authenticates with their OpenID Provider (OP). The user can then either elect to proceed or cancel authenication, depending on what kind of reputation is held by the RP. Assuming the user proceeds to use the RP, once they are finished with the service, the user is able to provide an evaluation of the RP which is then stored by their own OP. A subsequent user connecting to the same RP obtains a list of recent OPs from the RP, each of whom are queried about their users' feedback. This feedback is aggregated and displayed to the user, allowing them to cancel the authentication if required.

This approach is limited to only providing reputation information about RPs. Being able to provide a reputation for users and OPs would be useful, for example, as it would allow service providers to also avoid malicious users. Further, also being able to provide reputation for OPs would allow users to determine which OPs provide the best service. The obvious issue of trusting the RP to provide a list of recent OPs leaves this system open to malicious behaviour. In particular, an RP could maintain a list of its own OPs that always provide favourable reputation for the RP, and that it would always return as being the most recent. Reputation on each of the OPs might overcome a component of this issue.

The aggregation computation uses a dynamically interchangeable reputation computation engine. The intention is that the computation engine can be changed for another when conditions change, for example, as more feedback is available. This leaves the aggregation with the system, rather than the consumer of the information, and limits the flexibility of how the information could be utilised. If the raw reputation information were provided to the consumer, it could decide on the best reputation computation, allowing it to tailor the results to its own needs.

2.6 Summary

This chapter introduced a number of aspects that are important to this thesis: a terminology for reputation, two standard reputation models, a survey and taxonomy, an analysis to find research gaps and an examination of existing systems.

The standard reputation terminology and models for both reputation and reputation context are important as they allow for a consistent discussion of reputation and reputation systems within this thesis.

The survey of existing reputation systems was used to build and refine the reputation systems taxonomy. This taxonomy was used to classify a large number of academic and commercial reputation systems. The classification was used to find gaps in the existing research and literature. A subset of these gaps were selected as the research goals for this thesis.

Finally, a number of reputation systems that attempt to address some of the research goals were examined. None of the systems addressed all of the research goals.

Chapter 3

Architecture

GRAFT is an open and distributed framework that collects and makes available recommendation information about entities[1]. Recommendations about entities may include reputation, competency and demographic information. There is no centralised governance, with all participants in the framework required to contribute storage and bandwidth resources to a peer-to-peer network (P2P). Entities are typically real-world users, however they may also include electronic entities such as web services. Every entity in GRAFT is uniquely identified by their OpenID[2].

Figure 3.1 shows an overview of the major components of the GRAFT framework. At the centre, GRAFT nodes store all of the recommendation information about entities in profiles. Every entity has a single profile that may contain multiple recommendations (each from a different recommendation source). These profiles are replicated across a peer-to-peer network, ensuring that nodes leaving the network do not adversely affect the availability of profiles. Replication of profiles also limits the impact of malicious tampering, by making multiple copies available for comparison. A simple threshold vote can be used by a recommendation consumer to determine the most prevalent copy of a profile in the network.

Recommendation sources are the origin of recommendation information, both explicit and implicit. Explicit recommendations are those that were designed with the intent of representing reputation, such as "karma" and rating scores. Implicit recommendations are those that can be used as a stand-in for reputation, such as the number of friends in a social network. Recommendation sources include specialist websites (for example, auction or expert sites), databases and social networks. These sources push regular updates onto the network about those entities that they know about. The information that

[1]The work in this chapter is partially based on the published paper F. Hendrikx, K. Bubendorfer, Malleable access rights to establish and enable scientific collaboration, in: 9th International Conference on eScience, IEEE, Beijing, China, 2013.

[2]OpenID is further discussed in Section A.1.

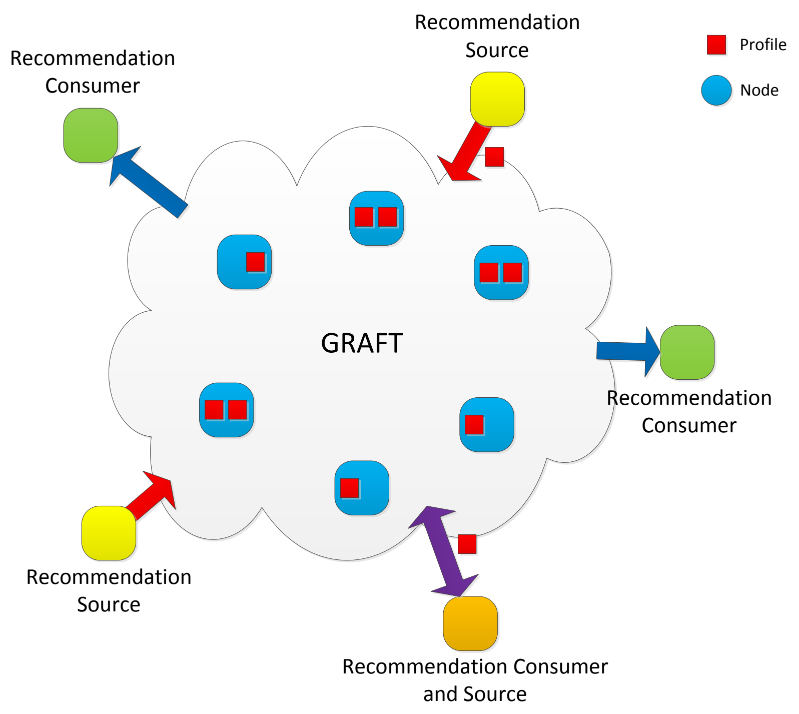

Figure 3.1: Overview of the major components of the GRAFT framework. At the centre, GRAFT nodes store all of the recommendation information about entities in profiles. Every entity has a single profile that may contain multiple recommendations, each from a different recommendation source.

is pushed by these sources is in the context of the source system and is not normalised. Recommendation sources are envisaged as being long-lived nodes, that are present the majority of the time.

Recommendation consumers utilise the information contained in profiles to make decisions. For example, a decision to continue with a transaction, or grant access to a resource. A recommendation consumer will fetch copies of the profiles of those entities in which it is interested. Policies maintained by the consumer are then utilised to evaluate the profiles. Recommendation consumers are intended to comprise nodes that are both short and long lived. In particular, some consumers may join the network simply to obtain profiles for the purpose of decision making.

3.1 Requirements

GRAFT was designed with eight requirements in mind. Section 1.3 discusses the key research goals that influenced the requirements. All of the requirements and their impacts on the design are discussed below. The influence of the research goals on the requirements is also noted where appropriate.

1. **Multiple sources of information:**

   (a) **Multiple sources:** GRAFT has been designed to allow multiple recommendation sources, some of which may even represent competing interests or approaches. This approach allows a recommendation consumer to pick the source that has the best reputation, or best fits its own requirements. For example, a recommendation consumer may pick a source based on locality rather than reputation.

   (b) **Use of derived sources:** GRAFT sources can be either explicit or implicit sources of reputation information. A number of examples of derived information sources including a source based on relationship information and a source that utilises academic-ranking information are presented in this thesis. *This requirement is based on research goal G.1.*

2. **Support for both humans and electronic entities:** GRAFT is designed to treat both humans and electronic services equally. In particular, the GRAFT profile was constructed in such a way as to support recommendation information for both classes of entities. Using OpenID as the unique identifier further supports this goal by making access to profiles identical in each instance. Electronic services are easily identified by their URL, which is the same as their OpenID. *This requirement is based on research goal G.2.*

3. **Integration of identity and reputation:** Every entity in GRAFT has an OpenID and as a direct result of that, a profile. Every entity can therefore be evaluated based on their prior feedback, including recommendation sources and consumers. This allows a recommendation consumer to pick a source from two competing sources based purely on their recommendations. *This requirement is based on research goal G.3.*

4. **Context support:** GRAFT has been designed to support the storing of contextual information. In particular, all recommendation information is enhanced with a source context. This context allows a consumer to determine the usefulness of a piece of information in relation to its own context. *This requirement is based on research goals G.4 and G.5.*

5. **Flexible:**

   (a) **Policy-based decision making:** Recommendation consumers utilise policies to process profiles and make decisions. The use of policies has a number of benefits, including rapid fabrication of new rules, straight-forward changes to existing rules, ability to share policies and little to no programming knowledge required to implement policy changes.

   (b) **Support for a variety of recommendation sources:** A wide-range of recommendation sources allow consumers to construct more nuanced policies that take into account a variety of factors when making decisions. A recommendation source that publishes information about an entity's occupation may allow a consumer to maintain, for example, policies that utilise chronological information to determine whether an entity is allowed access to a particular resource at a given time.

   (c) **No normalisation:** The information contained in the profiles is not normalised. This is because normalisation potentially limits the overall flexibility of the system, as it assumes a standard approach to rating and normalisation across all possible contexts. A normalised value may not make much sense outside of the original context, hence reducing its usefulness. *This requirement is based in part on research goals G.4 and G.5.*

6. **Scalable:**

   (a) **Scalable Architecture:** GRAFT has been designed to provide recommendation information in a number of scenarios. These range from acting as a drop-in replacement for existing reputation systems, through to providing a global recommendation system. In order to be able to scale up to the large sizes required, the architecture has been designed to be decentralised and distributed.

   (b) **Automated:** Profiles are maintained autonomously, requiring no human intervention or management. In particular, recommendation sources push regular updates to the network, ensuring that the profiles always have recent information. New recommendation sources may start operating in the network without any prior knowledge of existing entities or profiles.

   (c) **Decentralised:** GRAFT does not have any centralised governance or components. All participants in the network are members of a single peer-to-peer network, implemented using a Distributed Hash Table (DHT), that allows them to communicate and share profile information. As a result, bandwidth and storage are evenly spread and utilised throughout the network.

   (d) **Minimal resources:** A key reason for using a decentralised approach is that it reduces the infrastructure requirements. Many reputation systems are centralised and would require significant investment in order to scale them to a global size. Utilising a distributed approach ensures that bandwidth and storage requirements are evenly spread across participants in the network.

7. **Secure:** Although recommendations provide only a "hint" of an entity's character, and should only be used in support of other information, it is still sensible to limit the potential for fraudulent behaviour. To that end, GRAFT incorporates design decisions which help secure the recommendation information.

8. **Open and Standardised:** Where possible, GRAFT leverages existing technical standards, approaches and protocols in order to maximise interoperability and reduce cognitive overhead. In particular, GRAFT leverages OpenID to obtain a single, durable identifier for each entity in the system. GRAFT also extends the existing OpenID approach to OpenID provider (OP) discovery, and information exchange. *This requirement is based in part on research goals G.3 and G.5.*

3.2 Architecture Overview

A key design feature of the architecture is that it is fully distributed, with no centralised control or management. A consequence of this design decision is that profiles must be replicated multiple times, thus increasing the chance that a given profile will be available. The recommendation information in each profile is obtained from multiple sources. Sources include explicit reputation sources such as auction and forum websites, but may also include non-explicit sources of reputation, such as social networks.

Each source pushes regular updates about entities to GRAFT. An entity's OpenID is used as their key, allowing GRAFT to combine the information from each source into a single profile per entity. The continuous nature of updates to a profile means that they are dynamic and act only as a cache for an entity's recommendation information, and should therefore also not be thought of as belonging to the entity. For example, although an entity may own a particular OpenID, the profile that can be located in GRAFT using this same OpenID should be considered dynamic information that describes the entity at this point in time.

Figure 3.2 shows a high-level view of a profile with three recommendations. Each recommendation has a source, target and rating with context. The source is the OpenID of the source that generated the recommendation. Within a profile the target will always be the OpenID of the profile owner. The target is always included as this ensures that recommendations are complete within themselves. The rating is the "reputation score" for that entity in the context of the source. For example, for a source that represents a Forum, the rating may be an integer that represents the reputation of the target as a forum contributor. Access to entity profiles in GRAFT is possible via two distinct models:

1. **Lightweight Model:** This model extends OpenID and allows an entity to obtain and update target profiles using a web-services approach, similar to what is found

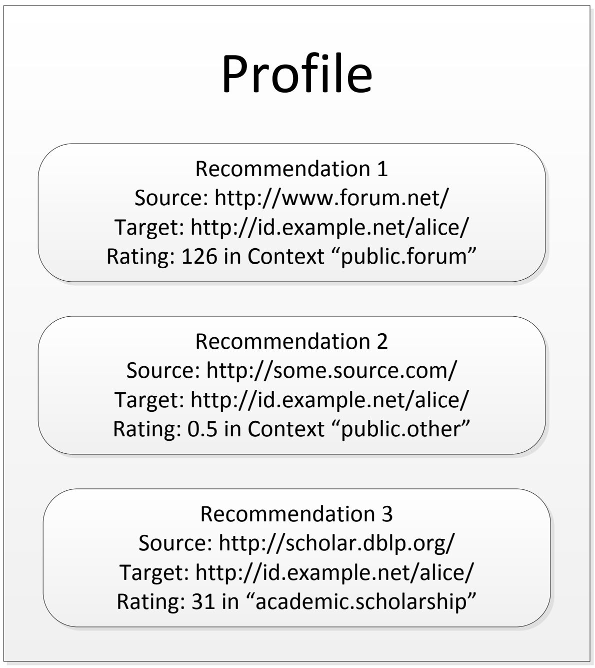
Profile
Recommendation 1
Source: http://www.forum.net/
Target: http://id.example.net/alice/
Rating: 126 in Context “public.forum”
Recommendation 2
Source: http://some.source.com/
Target: http://id.example.net/alice/
Rating: 0.5 in Context “public.other”
Recommendation 3
Source: http://scholar.dblp.org/
Target: http://id.example.net/alice/
Rating: 31 in “academic.scholarship”

Figure 3.2: A sample profile with some recommendation details shown. This profile contains three recommendations.

in OpenID itself. The focus of this model is on the distribution of profiles, and as such it is most suitable for web-based services that would like to augment their information about an entity with recommendations.

2. **Integrated Model:** This model includes all of the functionality from the lightweight model, but also allows an entity to verify target profiles. Profiles are replicated across a peer-to-peer network, allowing a consumer to determine the authoritative copy (using threshold voting).

From the perspective of simple, web-based information access, it is sufficient to implement only the lightweight model. However, this model affords no ability to verify a profile, so all results obtained using this model should be treated with caution. A more complete approach is presented by the integrated model, which ensures that profiles are verified prior to their use.

3.3 Lightweight Model

In the lightweight model, GRAFT leverages and extends the existing OpenID infrastructure to also include recommendation access, storage and retrieval. In particular, GRAFT providers (GP) are just like standard OpenID providers (OP), but instead of handling only OpenID authentication, GPs also handle GRAFT profiles.

3.3.1 Entities

Any entity connected to the Internet, whether human being or electronic service, is given equal status within GRAFT. Each entity must have an OpenID, and by implication, a GRAFT provider (GP). The usage of OpenID allows all GRAFT nodes to uniquely identify an entity, and locate its GP via standard discovery processes that leverage OpenID.

3.3.2 OpenID

With the recent release of OpenID Connect, OAuth 2.0 [133] has become the latest approach to authentication and authorisation for web-based applications. However, OAuth is not as interoperable as OpenID, as the specification actually describes a framework, rather than a protocol. Commercial federated identity offerings like Facebook Connect [366] are not sufficiently open to meet the design requirements for GRAFT.

OpenID was selected as the base-building block in GRAFT as it is a well documented and widely supported protocol for decentralised authentication. Key features of OpenID that enable the GRAFT framework include global uniqueness of each OpenID, the ability to dereference an OpenID in order to discover more information about an entity, and the fact that it is widely supported in existing software.

3.3.3 OpenRep

The extensions to OpenID to deal with GRAFT profiles are collectively called OpenRep[3]. OpenRep has two approaches to making profiles available for consumption.

In the first approach, a profile is discovered using the OpenID of the target. The profile can then be fetched using a RESTful [109] interface to the GP. This approach is similar to that discussed in OpenID, and is useful for obtaining profiles in a web-based context. This approach could be implemented as a simple way to fetch background information on an entity.

In the second approach, a profile is sent to a relying-party (RP) as the last step of the standard OpenID authentication process. Figure 3.3 outlines a typical OpenID authentication sequence [99] which has been augmented to include a GRAFT profile transfer from the GP to the RP. The profile transfer step (9) ensures that the RP receives a copy of the authenticating entity's profile. This copy is obtained using the Attribute eXchange (AX) extension [134] to OpenID. This extension is widely supported and is often used to transfer nickname or email address details to the RP. In this instance the AX extension is

[3]The technology used in OpenRep is discussed in further detail in Section A.2.

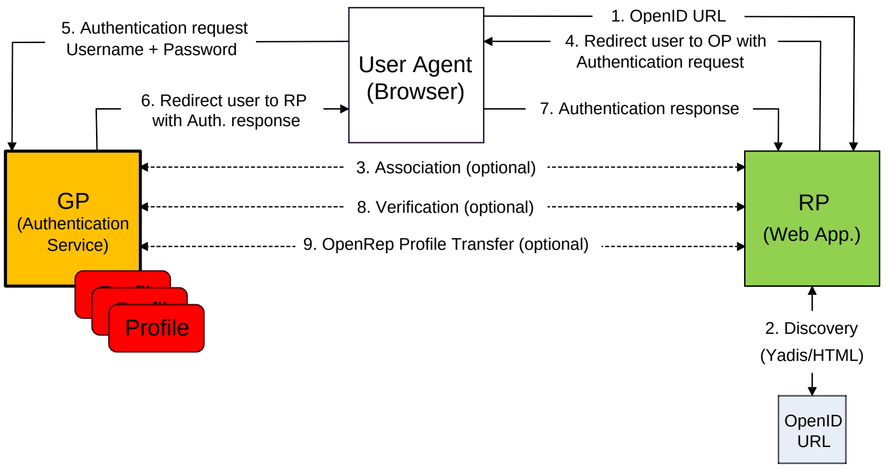
5. Authentication request
Username + Password
User Agent (Browser)
1. OpenID URL
4. Redirect user to OP with Authentication request
6. Redirect user to RP with Auth. response
7. Authentication response
GP (Authentication Service)
3. Association (optional)
8. Verification (optional)
9. OpenRep Profile Transfer (optional)
RP (Web App.)
Profile
2. Discovery (Yadis/HTML)
OpenID URL

Figure 3.3: OpenID Authentication Sequence [99], augmented with the profile transfer step.

used to transfer a profile with a single recommendation. The recommendation is the one previously generated by the RP, or one from another site, but in the same context. The latter case allows the RP to bootstrap an entity's reputation from a site with a similar context (for example, an expert forum could use an entity's reputation from another expert forum).

3.3.4 Analysis

The lightweight model is limited because it does not deal with the verification of profiles. All profiles are fetched directly from the GP, and because the GP can be self-hosted, the accuracy of a profile is potentially compromised. This model is therefore only really useful within limited and contained situations. Two such situations are described below.

In the first situation, a trusted GP makes available recommendation information to consumers within the same governance. This could be useful in situations where a number of consumers require the information, but do not want to implement the entire protocol stack. For example, an organisation that maintains its own GP and has the need to utilise recommendation information outside of that GP.

In the second situation, the information provided by a GP is useful during a login sequence. Section 3.3 describes how the current user's profile is transferred to the RP as the last step in a login sequence. This information, although not trustworthy, could be used to provide an initial "snapshot" of the user's reputation, weighted by the level of trust that the RP has in the GP. For example, a GP with a low level of trust from the RP's perspective could use a low weight such as 0.2, while a GP with a higher level of trust might have a weight of 0.8 applied to its reputation ratings. The RP could then obtain a validated profile via the integrated model as a background task, adjusting a user's reputation if it differs from the "snapshot" provided during login.

The initial development of GRAFT with the lightweight model supported both fetch and store operations for profiles held at a GP. The store operation should not be present for any public system as this provides an easy way to compromise the content of profiles.

3.4 Integrated Model

The integrated model extends the lightweight model, and was designed specifically to address the issue of profile veracity. By distributing multiple copies of a profile across a distributed hash table, the effects of malicious profile changes might be minimised or nullified.

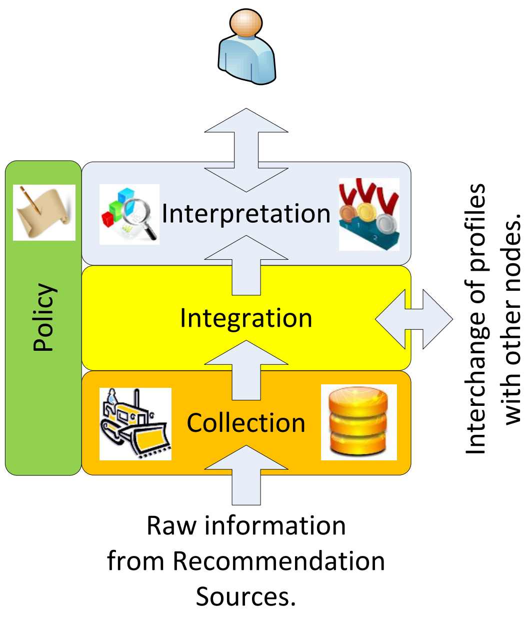
Policy
Interpretation
Integration
Collection
Interchange of profiles with other nodes.
Raw information from Recommendation Sources.

Figure 3.4: GRAFT Stack. Every GRAFT node is designed around a modular stack that comprises three layers, each with its own distinctive functions. A separate and extensible policy backbone enforces collection, integration and interpretation policies.

Within the integrated model, every application or system that intends to connect to GRAFT and utilise the profiles to their fullest extent is called a GRAFT node. Every GRAFT node must implement a minimal set of features from the GRAFT stack in order to be able to communicate with other GRAFT nodes, and provide their share of the resources required to support the GRAFT framework.

As shown in Figure 3.4, every GRAFT node is designed around a modular stack that comprises three layers, each with its own distinctive functions. A separate and extensible policy backbone enforces collection, integration and interpretation policies. The remainder of this chapter presents the three layers of the stack and the policy backbone, including detailed descriptions of the components and protocols that make up each layer:

1. **Collection Layer:** This layer ensures that raw information collected from an originating source is formatted into recommendations, and passed up to the next layer of the stack.
2. **Integration Layer:** This layer is responsible for each GRAFT node's integration into the overall peer-to-peer network. Recommendations may be received from either the collection layer, or from other GRAFT nodes. This layer may pass profiles up to the next layer of the stack.
3. **Interpretation Layer:** This layer is responsible for the verification, ranking and ordering of profiles.
4. **Policy Backbone:** The policy backbone enforces policies across the collection, integration and interpretation layers.

3.5 Collection Layer

The collection layer is responsible for obtaining raw information from, for example, an application, database or web scraper to which it is connected. These are called originating sources.

The connection to an originating source may be as simple as a database link, or the output of a process. Every unique originating source of information will require its own custom collection layer, with the output of this layer always being a recommendation. In most instances, the collection layer will need to define or generate metadata that describes the origin and context of the information.

Depending on the source, the raw information may be aggregated or simplified at this point, however no further interpretation of the information takes place at this layer. All information is formatted into recommendations and passed up to the next layer of the stack. The context in which the information was generated is always maintained, as this allows a recommendation consumer to make sense of the information later.

3.5.1 Entities

In order to allow a custom collection layer to create a valid recommendation for an entity at a given originating source, the entity must have provided an OpenID to the source, either by authenticating itself using OpenID, or providing a mapping from its local identity to its OpenID. Recommendations cannot be generated for entities without an OpenID, as GRAFT will have no way to uniquely identify that particular entity.

It is expected that entities will be able to turn on and off the export of their recommendation information from a source. For example, in the case of the source being a website, the entity should be able to login and choose to turn off the exporting of their recommendation information. However, providing an entity did not authenticate itself with OpenID, or provide a mapping to the source, this remains an "opt-in" exercise, as there is no way for a source to obtain this information otherwise.

3.5.2 Recommendations

Every recommendation can be thought of as describing a relationship from a recommendation source to a target entity. Each relationship can, in effect, be good, neutral or bad. Each recommendation is complete in itself, as individual recommendations may be exchanged between the different layers of the GRAFT stack. Every recommendation consists of a source and a target entity, and one or more assertions that describe either the relationship between the source and target, or an evaluation of the target. In either case, the assertion is made relative to the perspective and specific context of the source. For example, a source for an auction site may provide an assertion that relates to a user's rating in buying and selling at auctions. A seller may have additional assertions that indicate their honesty, level of service and shipping time for goods that they sold, while a buyer may have assertions that indicate reliability, and speed of payment.

In those cases where a recommendation source operates in more than one context, it may include multiple assertions, each with a unique context. The context is important as reputation information is context dependent [222, 325]. As discussed previously, assertions containing rating values will not be normalised. The context is also useful for bootstrapping the reputation of a target registering itself on a new site. The new site will be unable to find its own recommendation in the target's profile, but may instead try to find a recommendation from another site that is from the same, or similar context[4].

Depending on the recommendation source and the type of recommendation, an evaluation of a target may be summarised from a number of underlying transactions. The details of these transactions may also be included in the recommendation. This is useful in those cases where a component of each of the transactions is relevant to the overall evaluation of a target. For example, in transactions involving a monetary exchange, the value of the transaction may play an important part in determining its significance [261]. A transaction with a value of \$1000 is more significant than one of \$5, and should there-

[4]The enumeration and construction of a taxonomy of contexts has been left as future work.

fore be weighted more heavily when making decisions[5].

3.5.3 Originating Sources

The originating sources from which GRAFT provides recommendations are varied. Recommendations are formed from raw source information that is collected from applications, databases and interfaces. These sources range from those with explicit reputation information, through to implicit and non-traditional sources.

Explicit reputation sources include sites that implement reputation mechanisms using approaches such as ratings, karma, votes and user feedback. These approaches are widely implemented on the Internet, with many sites using one or more variants to support their communities.

Implicit reputation sources include sites that have not defined a formal reputation system, however reputation information is still utilised by the community to assist them in making decisions. Social networking sites can be considered to utilise an implicit reputation system, as members can extract a degree of trust from the information obtained from friends, or friends of friends.

Any corrections to the information obtained from a source should be made by the originating source. This same approach to information is widely utilised in the real-world, where corrections must be made at the source of the information. For example, if the bank generates an incorrect credit score for a given customer, then it is up to that customer to contact the bank and have this corrected.

Figure 3.5 shows two examples of how the collection layer could be implemented for different originating sources. In the first example, the source has been granted direct access to the database of the originating application. The collection layer is able to see certain tables within the database and is therefore able to query these as required. This particular collection layer might be implemented as a series of SQL queries by someone with knowledge of the database. This source would only require minimal maintenance, as database structures are unlikely to change rapidly. Further, this source will always have access to all relevant information, and will therefore always be up to date.

In the second example, a web scraper that is unaffiliated with the application must utilise the publicly available web interface in order to download appropriate web pages via HTTP or HTTPS. This source is not as efficient as the first, as it requires regular maintenance to ensure that updates to the web interface do not adversely impact on its ability to find content. In addition to maintenance, there are other issues that impact on the efficiency of this approach. Firstly, the source must consider information recency, as in-

[5]eBay does not make this distinction however, as all transactions are weighted equally, regardless of their value. This allows sellers to artificially increase their reputation by selling lots of small value items, and then defaulting on high value transactions.

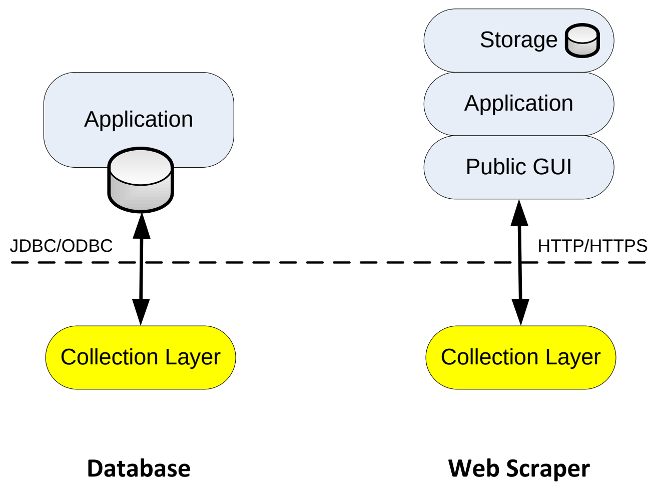

Figure 3.5: Database and Web Scraper raw sources. Two examples of how the collection layer could be implemented for different originating sources. In the first example, the source has been granted direct access to the database of the originating application. In the second example, a web scraper that is unaffiliated with the application must utilise the publicly available web interface in order to download appropriate web pages via HTTP or HTTPS.

formation may change between visits from the web scraper. Secondly, visibility must be considered, as public access may limit what the source can see. Finally, long-term access can be problematic, as web scraping is often considered a grey area and may eventually cause the web scraper to be locked out.

3.5.4 Recommendation Sources

Figure 3.6 shows a generalised sequence of events that a recommendation source must follow in order to update a recommendation within a profile. After authenticating itself with the originating source, the source uses the collection layer to fetch the latest information for the entity it is currently interested in. At this point the source will form a new recommendation. The source then fetches all of the profiles for this entity, using the integration layer. Once all of the profiles have been fetched, a threshold vote in the interpretation layer is used to determine the best profile. The recommendation that was formed earlier from the raw information is now inserted into the profile, and the profile is pushed back into the network.

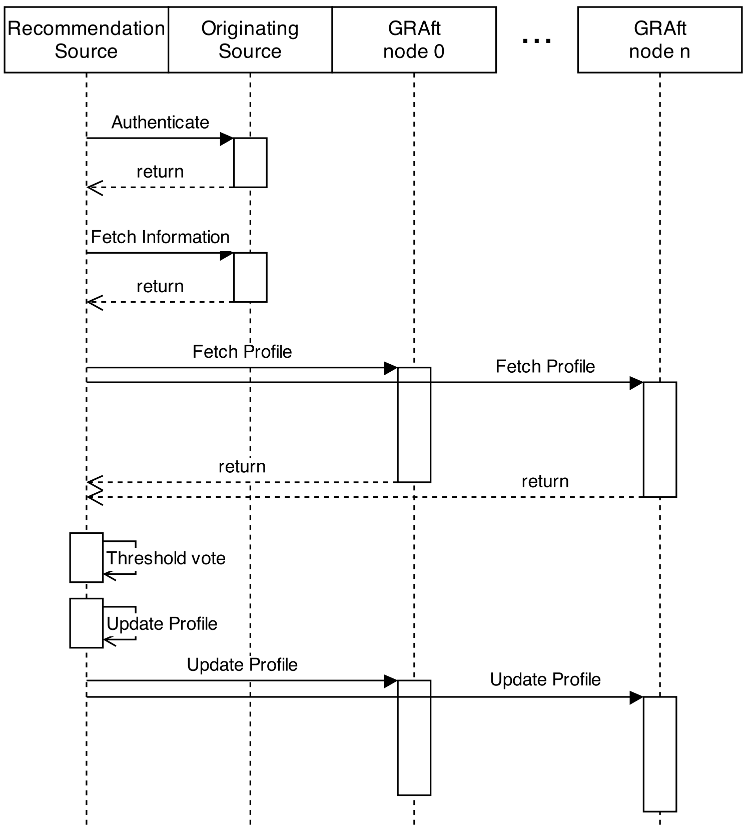

Figure 3.6: Recommendation Source Update Sequence. The generalised sequence of events that a recommendation source must follow in order to update a recommendation within a profile.

3.5.5 Analysis

The collection layer deals exclusively with raw information obtained from originating sources. On initial inspection there are no apparent issues, however it is important to remember that this layer is responsible for the generation of all recommendations within GRAFT. To this end, a number of issues must be handled by this layer:

1. Securing the interfaces and applications
2. Securing the recommendations
3. Intent of originating source

Securing the Interfaces and Applications

The security of the originating source interfaces and applications is essential, as a compromise in these systems will allow an attacker to control the recommendations that are generated from the data held by these systems. This particular point cannot be addressed by a design change in GRAFT, and must be handled during the development, deployment and ongoing maintenance of the collection layer for an originating source.

Securing the Recommendations

A standard approach to securing the content of a digital object, or in this instance a profile in the form of an XML document, is to sign it using an XML signature[6]. As GRAFT distributes only profiles, any signatures across the profile must form an inherent component of the profile itself. Further, as each recommendation comes from a unique source, each source must individually sign its own recommendations, rather than the whole profile.

These design constraints mean that each recommendation is signed by its source using an enveloped signature (as opposed to an attached signature, which would be distributed in a separate file). As shown in Figure 3.7, having a single signature per recommendation allows a receiver to confirm that a given recommendation is accurate and has not been modified. In particular, XML signature protects signed content against both simple modifications and attacks where the content and checksum have both modified in an attempt to hide the changes.

XML signature also provides a receiver with the ability to authenticate the signer. Since the signer identity within GRAFT is the same as the source identity, it is a simple matter to confirm that the source and the signer are the same. Confirmation of the source identity can be achieved by checking the public key of the source against the signature on the recommendation.

[6]`http://www.w3.org/TR/xmldsig-core/` - last accessed October 2014.

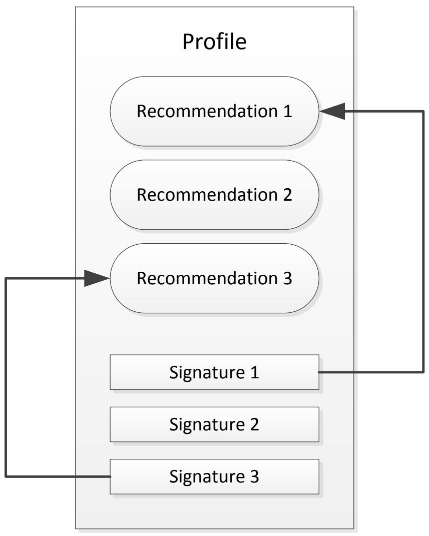

Figure 3.7: Enveloped Signatures in the GRAFT Profile. A single signature per recommendation allows a receiver to confirm that a given recommendation is accurate and has not been modified.

Intent of Originating Source

It is assumed that all sources joining GRAFT have good intentions. However, it is possible that a source may join the network with bad intentions. In particular, a source may generate false recommendations about individuals, attempt to corrupt profiles by changing recommendations, or remove some or all recommendations from a profile.

The first case, where a source generates false recommendations, is handled by examining the other recommendations in a profile, and by examining the recommendation profile of the source itself. In particular, a source with few, or poor recommendations should be considered suspect. In the case of profile corruptions, these changes would be detected using the XML signatures. Finally, in the case of recommendation removals, the profile would gradually "recover" as sources updated their recommendations over time. At worst, an entity might have missing recommendations until the majority of sources had updated their recommendations.

3.6 Integration Layer

The integration layer receives recommendations from the collection layer and profiles from other GRAFT nodes. Recommendations from the collection layer may be integrated into new or existing profiles, before being pushed up the stack to the interpretation layer, or pushed back into the network. This layer is mandatory for all GRAFT nodes, as it is this layer that permits the exchange of recommendation information between nodes. In particular, this layer implements a peer-to-peer network (using a Distributed Hash Table[7]) that underlies the entire network.

3.6.1 Profiles

Every entity in GRAFT has a single profile potentially made up of multiple recommendations. This profile is constructed using XML, which allows humans to easily read the profile, and allows for machine parsing. Further, text based formats such as XML can be easily transferred over HTTP and HTTPS, which is an important consideration given the potential uses of the GRAFT framework.

A timestamp field can be used to compare two profiles to decide which is newer, while a counter records the number of recommendations in a profile. Each of the recommendations in a profile is indexed by a unique key. This key is always the recommendation source's OpenID. This allows a source to easily find its own recommendation in a profile, and ensures a one-to-one mapping of sources to recommendations.

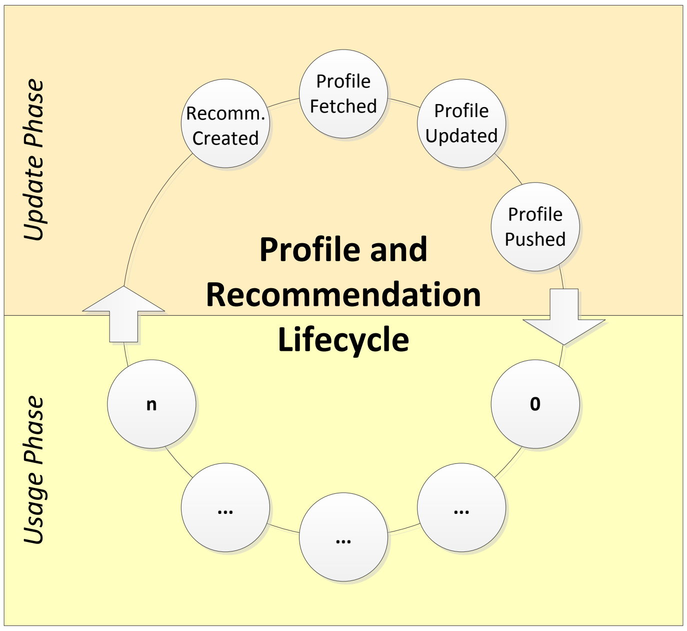

Figure 3.8: Profile and Recommendations Lifecycle. This figure illustrates the lifecycle of a profile and the recommendations contained within it.

Figure 3.8 illustrates the lifecycle of a profile and the recommendations contained

[7]The concept of distributed hash tables is further discussed in Section A.3.

within it. In particular, the lifecycle has two distinct phases: update and usage. In the update phase, a recommendation is first created by the recommendation source. The profile is then fetched from the network and updated with the new recommendation. The profile is then pushed back into the network. In the usage phase, the profile is fetched and recommendations within it used by consumers. For example, a profile may be fetched and consumed during login, when a consumer wants to check the recommendations held by the authenticating entity.

3.6.2 Concurrency

A key assumption of the architecture is that profile updates from every source occur regularly. However, the time between updates may be measured in days or weeks. In particular, for those sources that represent many entities, or require intensive computation, the time between updates may be long.

The architecture does not explicitly consider the impact of concurrency issues on profiles, leaving the outcome of simultaneous writes to the same profile entirely to chance. For example, two sources attemping to update a profile at the same time will lead to unexpected results in terms of the profiles that are stored in the network.

However, the integrated model provides the architecture with an approach for dealing with the outcome of this situation: threshold voting. By selecting the most prevalent copy of a profile, the architecture already has a mechanism to decide which is the authoritative profile. Unfortunately, churn and malicious nodes reduce the chance that any profile will reach a threshold. Rather than burdening the threshold voting mechanism with yet more issues, future research needs to consider either the introduction of a concurrency mechanism, or seek to avoid the issue altogether. These two options are discussed further below:

Concurrency Mechanism

Concurrency mechanisms come into two essential types: Optimistic and Pessimistic. Optimistic mechanisms assume that a transaction can be progressed until it is time to commit, at which time it can be aborted if anything is wrong. For example, in the GRAFT context, a profile could be updated with a new recommendation. However, pushing this profile back to the network could be aborted if another update transaction was found to be in progress. Pessimistic mechanisms try to avoid these situations by putting safeguards in place that prevent undesirable outcomes. For example, in the context of GRAFT, a pessimistic mechanism would seek to lock a profile before an update was allowed to take place. As a result, pessimistic mechanisms do not perform as well as optimistic mechanisms.

An optimistic mechanism would be much better suited to GRAFT and not hamper its performance significantly. However, this does require the addition of an approach that allows a node to detect an "in progress update" to a profile. This exercise is left for future research.

Avoiding Concurrency Issues

Reducing the granularity of information within GRAFT from profiles down to individual recommendations would remove the need for any concurrency mechanism. This is because each recommendation comes from a unique source, and only that source will ever update its own recommendations, thus allowing simultaneous writes to the profile. This is clearly a desirable situation, however the larger problem in this instance would be the ability to efficiently locate recommendations, and use that information in the construction of a complete profile. This is also left as future work.

3.6.3 Overlay Network

Every GRAFT node belongs to a single peer-to-peer network implemented using the Kademlia distributed hash table. Kademlia[8] was selected as the Distributed Hash Table (DHT) implementation because it is well researched and supported in the scientific community. In addition, many of the widely deployed overlay networks in use on the Internet are based on Kademlia [41], and it has proven robust against certain types of attacks.

The DHT provides GRAFT nodes with the ability to store and retrieve key/value pairs that are distributed over the nodes in the network. This is used by GRAFT to store multiple profile replicas, using the OpenID of the entity as the key. The built-in Kademlia key/value pair replication is not used in GRAFT because sources regularly fetch and write updates to profiles, obviating the need for additional replication. Figure 3.9 shows two sources, two providers and two consumers as GRAFT nodes in the DHT. The source nodes are each the point of origin of a single item of recommendation information within a profile. Each profile will contain entries from many sources.

3.6.4 Analysis

The integration layer implements the distributed hash table upon which much of the GRAFT functionality depends. A distributed hash table is a form of overlay network, in that it is built on top of the TCP/IP network that comprises the Internet. Urdaneta et al. [309] define three key attacks on overlay networks:

[8]Further information about Kademlia can be found in Section A.3.1.

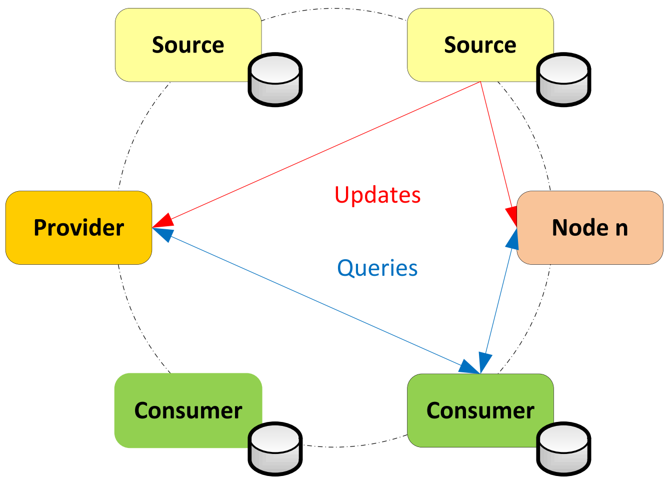

Figure 3.9: DHT Structure. Two sources, two providers and two consumers as GRAFT nodes in the DHT. The source nodes are each the point of origin of a single item of recommendation information within a profile.

1. **Sybil attack:** an attacker creates a large number of nodes in the network, until the attacker controls a fraction of all nodes. The aim of this attack is the introduction of malicious nodes into the network.
2. **Eclipse attack:** an attacker tries to isolate a benign node, such that all messages to or from this node are routed over a malicious node. The aim in this attack is the isolation of a node, or series of nodes from the main network.
3. **Routing and Storage attacks:** where malicious nodes subvert routing protocols and corrupt stored data. The aim of this stack is the modification of node behaviour and data in the network.

Sybil attack

In a Sybil attack, an attacker attempts to gain an overly large influence over a peer-to-peer network by creating many nodes. These nodes are all controlled by the attacker and can act in unison to modify communications, or make changes to information being routed over part of the overlay. As a peer-to-peer system is distributed, it is easy to create many new nodes as there is no centralised control over resources. An attacker can continue to create nodes until they control some fraction of all nodes in the network [41]. Douceur [96] proved that this attack cannot be prevented, but only limited.

In centralised systems a limit might be placed on new nodes by requiring that they are “bound” to an individual, or that they pay some kind of monetary fee [41]. For distributed systems, a number of solutions have been presented that attempt to address this issue. These solutions can be grouped into a number of classes. The following is loosely based on the classes presented in [309]:

- **Centralised certification**: New node identifiers are issued by a central authority. An application for a new identifier is checked by this authority to ensure that the requestor has not previously obtained an identifier. This makes the central authority a key target for attacks, but potentially also limits the scalability of the network. A certification system that uses a secure protocol and authentication between nodes is discussed by Aiello [28]. Similarly, Singh [290] uses cryptographic identifiers.
- **Distributed registration**: Similar to centralised certification, this approach attempts to maintain a number of registers. New nodes register themselves with a register, which tries to limit the number of nodes that may be registered at a given IP address. This approach is limited by the fact that it relies on the DHT itself to maintain the registers and is therefore open to all of the same attacks.
- **Physical network characteristics**: Physical network characteristics are assumed to be difficult to fake, and therefore new node identifiers are based on these characteristics, or the characteristics are measured to try and detect multiple nodes from the same source. Measuring network characteristics is transparent to the nodes, however the required infrastructure is itself vulnerable to attack. Assigning a new identifier based on a node’s IP address is attractive, however this limits every host to a single node and does not deal with NAT.
- **Social networks**: Yu et al. [345] propose SybilLimit, an approach that utilises social networks to provide resilience against Sybil attacks. Lesniewski-Laas [190] proposes a similar approach using one-hop networks (where friends are within a single hop of each other).
- **Computational Puzzles**: Nodes must periodically validate their identity using a computational puzzle, with the goal that completing these puzzles requires significant computational resources and as such an attacker would have trouble maintaining their multiple identities. This approach does not work as it assumes a homogeneity in computing resources. A crypto-puzzle approach, along with multiple disjoint lookup paths is discussed by Baumgart et al. [41].
- **Game Theory**: As with the centralised approach, this solution requires nodes to pay a “monetary fee” to join the network. Aside from the difficulties in implementing such a solution, a distributed currency would potentially be open to the same issues as the DHT overlay.

- **Reputation**: This approach promotes the idea that the reputation information already held in GRAFT could be used to limit the creation of new nodes. This could be accomplished by linking the creation of new nodes in Kademlia to an OpenID, and therefore a profile. Profiles with a greater overall reputation should be able to create more nodes, and each node should receive a share of the overall reputation of the creator. In other words, the number of new nodes that can be created should be proportional to the reputation of the creator. Poor behaviour by any of the nodes that were created would be reflected back on the creator.

As shown by Douceur [96], this attack cannot be prevented, and as such any solution selected needs to weigh the costs of its implementation against its potential effectiveness. For example, as GRAFT's design goals include scalability, restricting node identifiers to IP addresses would limit its usefulness on the Internet given the prevalence of NAT and CGN (Carrier Grade NAT). In a similar vein, introducing centralised certification or registration services simply limits the ability for the architecture to scale.

Given its use of the Kademlia distributed hash table, GRAFT is currently open to Sybil attacks. As with many peer-to-peer systems, GRAFT relies on the existence of multiple nodes and data redundancy to ameliorate the effects of malicious nodes [96, 340].

Future research needs to consider the function and operation of GRAFT. In particular, using the reputation information available within GRAFT to limit the effectiveness of attacks needs to be considered. As noted above, it might be possible to use the reputation of an entity to limit the number of new nodes that it may create. Further, as GRAFT may have a number of long-lived nodes in the form of the source nodes, adopting an approach that uses the long-lived nodes to control a form of distributed registration may also work for GRAFT.

Eclipse attack

In an Eclipse attack, the attacker attempts to control a sufficiently large proportion of nodes such that any target nodes are "eclipsed", or cut-off from the rest of the network [103]. In effect, all information to and from the target nodes must transition over an attacker's nodes. This attack, also known as route-table poisoning [309], is related to the Sybil attack, but is possible even in cases where the Sybil attack has been prevented [290]. A number of solutions have been proposed. The following approaches are loosely based on the solutions presented in [309]:

- **Induced churn**: Subsets of nodes are assigned new identifiers once per "period", making it harder for attackers. This introduces a lot of additional overhead into the system, and depends heavily on a centralised "randomness" server.

- **Region-based routing tables and induced churn**: New nodes are assigned identifiers based on consensus, further, all nodes in that "region" of the identifier space must leave and rejoin with new identifiers.

- **Constrained routing tables**: Strong constraints are imposed upon a second routing table that only contains verified information [62].

- **Connectivity Auditing**: Nodes anonymously audit each other's connectivity. Nodes found to have a significantly higher "in-degree" than the average may be mounting an attack and should thus be avoided by other nodes [290].

Urdaneta et al. [309] discuss how there is no single best approach to securing a DHT, and note that the exact approach selected to security is best determined by the needs of the application using the DHT. For example, PlanetLab uses OpenDHT, which because it does not generally need to be concerned with NATs uses hashed IP addresses for node identifiers. Similar to the previous section, GRAFT is open to eclipse attacks because of its use of the Kademlia DHT. Eclipse attacks are often mitigated using replication and "self-authenticated content" [290]. The solutions presented in the previous section may help to mitigate some of the effect of eclipse attacks, however further research is required.

Routing and Storage attacks

After the creation of malicious nodes using the sybil attack, and the isolation of target nodes using an eclipse attack, routing and storage attacks can proceed to damage the DHT. In routing attacks, the goal is for malicious nodes to disrupt routing in the network by, for example, failing to forward a lookup request, or forwarding a request incorrectly. Storage attacks are similar, however the focus is on providing tainted responses, for example, a malicious node may pretend to be responsible for a given key when it is not. According to Urdaneta et al. [309], defences against storage and routing attacks are based on two approaches:

- **Redundant storage**: Data replication is used to achieve storage redundancy. Multiple copies of the data are stored in the network, either at locations that are numerically close to the original key, or spread over the identifier space.

- **Redundant routing**: Messages to nodes are sent over multiple routes, or sent to multiple nodes simultaneously.

Unlike the previous sections describing sybil and eclipse attacks, GRAFT is not open to routing and storage attacks because Kademlia implements some features that are hard to attack. In particular, the routing table in a Kademlia node is hard to attack as the protocol tries to keep only highly available peers in the table [309]. Additionally, node lookup in Kademlia uses redundant routing, and GRAFT implements a form of redundant storage when storing profiles.

3.7 Interpretation Layer

The interpretation layer supports consumer decision making by ensuring that profiles have been fetched and verified, and that these are ranked and ordered in a meaningful way. Although fetching profiles is handled by the integration layer, the verification of these profiles is handled by the interpretation layer.

Profile verification is accomplished by conducting a threshold vote. Each profile should have at least p copies in the network. If m of p copies are the same, such that $m \geq t$, where t is the threshold, then we can be confident that the given profile is most prevalent in the network. The natural value for the threshold is just over $1/2$, however using a higher value such as $2/3$ could be used to increase confidence in the result as more copies must be in agreement.

This layer is typically only implemented by recommendation consumers such as ranking or reporting services, however an interpretation layer may also be used by service providers to evaluate the reputation of a service consumer. This allows service providers to avoid some types of denial of service attacks, by avoiding those consumers that have poor recommendations.

3.7.1 Analysis

Profile Replication

As discussed in Chapter 3, profiles are replicated across the peer-to-peer network. For each profile, p different hash functions are applied to the OpenID, allowing GRAFT to store and retrieve p profile copies, but using only a single OpenID as the key. When fetching a profile, a GRAFT node simply hashes the OpenID in the same p ways, and is then able to access all of the copies stored in the peer-to-peer network. The selected hashing algorithms must be able to distribute the keys in a uniform way, and generate 160-bit hash values, as these match the default SHA algorithm used in Kademlia. Given the random distribution of profiles across the network, there is little chance of duplicity. However, threshold voting may be used to verify any given profile.

Threshold Voting

It was previously discussed how threshold voting can be used to determine the most prevalent copy of the profile in the network. The digital signing of recommendations at the collection layer prevents modifications to the individual recommendations, and threshold voting ensures that attempts to add or remove recommendations are detected. This is important, for example, as a GP has full access to modify the profiles of any entities registered with it. As an entity can host their own GP, it has the ability to whitewash its own profile by removing certain recommendations.

3.8 Policy Backbone

The policy backbone is common to all three layers of the stack and is used to enforce policy at each of these layers. Policies guide decisions, and help to drive consistent decision making.

Within the collection layer, the policy backbone enforces how information is collected from a source. In particular, policies at this level control the frequency of collection updates, the age and granularity of the information collected and stored in a recommendation, and if any summarisation should take place. The policies at this layer are also responsible for enforcing an entity's privacy; it is assumed that every entity wishing to participate has an OpenID and has set a flag that announces their willingness to have their recommendation information exported from a particular site.

At the integration layer, policy is used to enforce which GRAFT nodes the current node may connect to. However, policy also enforces the reverse situation, where a GRAFT node may reject incoming connections. This allows a GRAFT node to avoid those other GRAFT nodes that are deemed to have a poor reputation.

Within the interpretation layer, policy is used to control profile verification using threshold voting. The number of profile copies p is determined by a global GRAFT setting, however the number that are actually fetched, and the threshold value t are both controlled by policies. The number of profile copies fetched is defined as an integer, while the threshold t is defined as either a ratio or percentage of profiles that must match in order for a profile to be seen as being correct.

Chapter 4

Case Studies

This chapter examines a number of case studies that explore how the GRAFT framework might be used. The case studies are organised into three sections: Access Control[1], Composition and Acquisition[2]. Each is explained in further detail below.

- **Access Control:** the process that either allows or denies access to a resource based on the recommendations held by an access seeker.
- **Composition:** the process used to compose a higher-level web service from a number of lower-level services, using the recommendation information for each service.
- **Acquisition:** the process used by an access seeker to acquire the rights to utilise a web service.

4.1 Access Control

The Internet has seen significant growth over the past decade [232, 76], and even the more modest growth rates seen recently [75] point to an increasing number of participants and usage. Since the advent of Web 2.0, the trend is towards using collaborative systems to meet, organise, discuss and cooperate. Within an organisation this improves communications and decision making, and reduces both time and cost [31] as it allows an organisation to apply the best people to any given situation. Collaborative computing (hence collaborative systems) encompass "the use of computers to support coordination

[1]The work in this section is largely taken from the published paper F. Hendrikx, K. Bubendorfer, Malleable access rights to establish and enable scientific collaboration, in: 9th International Conference on eScience, IEEE, Beijing, China, 2013.

[2]Some of the work in this section is taken from the published paper F. Hendrikx, K. Bubendorfer, Policy derived access rights in the social cloud, in: 9th International Conference on eScience, IEEE, Beijing, China, 2013.

and cooperation of two or more people who attempt to perform a task or solve a problem together" [31].

As with any computer system, collaborative systems require access control (and auditing) to prevent unauthorised access and change. Access control is defined in [275] as a process that requires "every access to a system and its resources be controlled and that all and only authorised accesses can take place". Access control policies can be classified into four groups: Discretionary, Mandatory, Role-Based [275, 276] and Attribute-Based.

- **Discretionary Access Control** (DAC) policies restrict access to an object based on the identity of the current user and a set of rules [276]. These systems typically assume that all objects have an "owner" [107], and users may reassign their privileges to others [275]. Privileges to an object are assigned to users when they create an object, and can be changed or revoked by an administrator.
- **Mandatory Access Control** (MAC) policies restrict access to objects using a global policy. The policy is centrally controlled by an administrator, and cannot be changed by the users. Every user and object is given a security classification. Users must have the same or higher security in order to read an object with a given security classification [276].
- **Role-Based Access Control** (RBAC) policies are based on the concept that access to an object is limited to certain roles that have been defined within an organisation. The responsibilities and qualifications of any particular user determine the roles that are assigned to them [275, 107, 276] and hence which objects they may access.
- **Attribute-Based Access Control** (ABAC) offers an approach that is well suited to service-oriented environments. In ABAC, all access decisions are based on characteristics of three elements: the requestor, the resource being requested and the environment [348, 323, 252]. Despite this, it should be noted that there is no widely accepted model for ABAC and only initial work in that direction [156].

A number of collaborative services are defined by Bafoutsou and Mentzas [31]. This section will examine and utilise two of those services: Forums (called Bulletin boards in the original paper) and File and document sharing.

- **Forums** allow for online discussions and interactions between participants with a similar interest. The main drive for a forum is the exchange of ideas, whether it be for problem solving, discussing politics or collaborative work between academics. All posts to a forum are effectively read-only, with each participant adding new content to the discussion. Off-topic posts can be ignored, while offensive ones can simply be removed by a moderator. Forum access controls are often based on a simple RBAC scheme: users acquire a *participant* role, which allows them to take part in the forum. A separate *administator* role is assigned to the forum owner, who maintains the forum. This user may promote other users by assigning them a *moderator* role.

- At its most basic, **File and document sharing** allows for the simple sharing of files between users. The users work with the files on their own equipment [31]. At its most complex, file and document sharing may be implemented using collaborative authoring system like a wiki. A wiki is an "interactive website" that allows users to view and edit their document content as pages [254]. All pages in a wiki are theoretically editable by any authenticated user [181]. All changes to a document build upon previous changes, allowing users to rapidly increase and enhance their content. Wiki access controls are usually based on a simple RBAC scheme, similar to forums: users acquire a *participant* role, which allows them to edit pages in the wiki. A separate *administator* role is assigned to the wiki maintainers.

There are a number of issues that may be identified in these collaborative systems, whether they use DAC, MAC, RBAC or ABAC access control policies. Some of these issues are:

- **Costly role analysis and definition**. RBAC requires a costly initial phase to identify and define an appropriate set of roles [310]. The definition of roles and permissions can sometimes lead to what has been called "role explosion" [185].
- **Maintenance of users, objects, roles and attributes**. DAC, MAC, RBAC and ABAC all require regular and ongoing maintenance of users, objects, roles and attributes.
- **Low level of flexibility**. DAC policies typically implement only a coarse level of access to objects, restricting what is possible. RBAC systems are slow to adapt in fast changing environments, and may not support policies that have dynamic components such as the time of day [185].
- **Tight coupling of access control to systems**. Access control is often hard-coded into systems, limiting both their flexibility and extensibility.
- **Standardisation**. In order to be useful, access control models must be well understood and standardised. Although implementations and standards do exist for ABAC, it has been argued that there is no consensus on what features ABAC should have and what the model should look like [156].

As discussed, these approaches all require ongoing administration and maintenance. In the case of RBAC, a costly initial analysis is required when starting with a new system, while DAC and MAC both offer limited flexibility to their organisations. Further, tight coupling of access control to systems limits their overall flexibility and extensibility.

In this section the case studies propose a new approach based on the user's demographics and behaviour, rather than simply their identity. Information derived from an entity's attributes and past actions are utilised by access control policies for decision making. At the simplest level, a user's rating from one website can be used at another, while a more complex case study involves obtaining information about a user from multiple sources and combining these. Since the information about the users is automatically derived from one or more sources external to the consumer, any changes to the user at these sources is made available to the consumer and reflected in the access control decisions that are made. This obviates the need for regular maintenance.

4.1.1 Forum

Web-based forums are a standard way for interest groups to communicate and exchange information on the Internet. In most cases, the forums are broken into multiple subforums, each with a number of discussions called threads. A thread may contain a number of messages in response to the initial message which started the thread.

A common feature present in many web-based forums is the ability to "vote up" a given message. A vote may increase the visibility of the message, or confer additional "karma" on the author of the message. A good example of this approach is SlashDot [13], where authors strive to maintain "good karma", by posting insightful or meaningful messages in the threads. In those instances where additional karma is conferred on the author, the amount of karma, or karma score, held by that author may, for example, give his or her new messages a higher default visibility, or grant the author additional rights within the forum. The latter approach is used within Stackoverflow [14], where users obtain further privileges as they gain reputation. Examples of additional rights include the ability to comment on postings, or the ability to vote on the relevancy of postings.

In this case study, the GRAFT framework re-implements the karma feature found in many forums. GRAFT is used to capture recommendation information from a web-based source forum, and make it available to a web-based consumer forum. In the simplest implementation, a single forum could be both the source and consumer of the recommendation information. However, in this case study it is considered from the perspective of two separate forums, with the consumer forum using GRAFT to store and bootstrap its karma scores. The goals of the case study are twofold: show how GRAFT can automate the maintenance of user profiles and access control policies, and how it can be used to implement existing features such as the karma scores used on SlashDot.

In the source forum, participants use a local pseudonym, but are authenticated using their OpenID. Assuming they have opted into using GRAFT, the collection layer is able

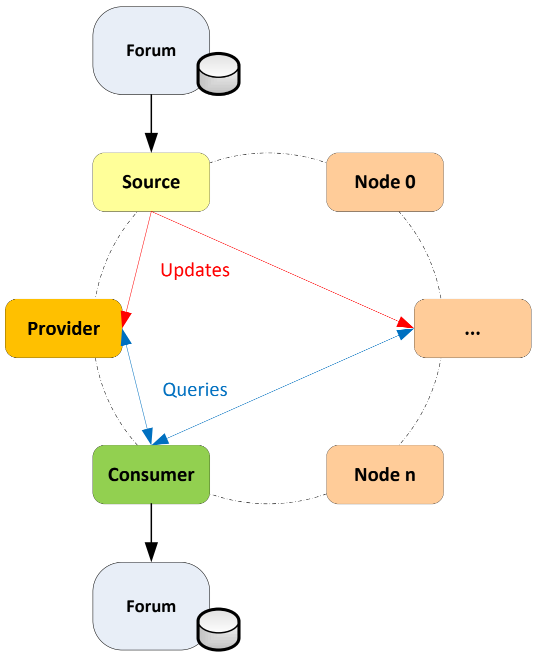

Figure 4.1: GRAFT enabled Forums. Recommendation information flows from the source forum to both the GP and the network.

to obtain their current karma score from the local database, and export this information to GRAFT. As shown in Figure 4.1, recommendation information flows from the source forum to both the GP and the network.

At a more detailed level, the source is able to map the local pseudonym to an OpenID, either because an OpenID was used for authentication, or because a mapping was provided by the user. The source uses the OpenID to retrieve the profile for the participant from GRAFT. The source then updates the profile to include a recommendation from itself. This recommendation contains the karma score of the participant, and the context for the source forum. The modified profile is then pushed back into the network.

Upon authentication of a participant, the consumer forum will be able to use either the lightweight model to receive a profile, or the integrated model to receive a verified profile. Assuming it was able to receive a profile, it will then search for a recommendation that has its own OpenID as the key. Recommendations are found within a profile using a unique key, which is always the OpenID of the source. In the case of a new participant it will be unable to find this recommendation, and may therefore choose to look for another recommendation, but in the same context as itself. This last feature allows a new participant to bootstrap their karma score at the consumer forum from the source forum, assuming they share the same context and that the consumer forum is willing to trust the source of the recommendation. The consumer forum could determine this by examining the profile of the source forum.

Policy

The policies in the consumer forum grant additional access rights to participants based on their recommendations. As described earlier, the karma rating stored within an appropriate recommendation could be used by the consumer forum to grant access to a feature, but only if it exceeds a given threshold.

$$r \geq t \tag{4.1}$$

Equation 4.1 describes this threshold relationship. If the rating r exceeds threshold t, then access will be granted to the feature or resource.

4.1.2 Wiki

The current situation faced by scientists is one in which they are required by funding bodies and the changes to endemic scientific practice to share and preserve their data. Kaye et al. [173] state that "Funding bodies have recently introduced a requirement that data sharing must be a consideration of all funding applications in genomics. As with all new developments this condition has had an impact on scientific practice, particularly in the area of publishing and in the conduct of research".

Tenopir et al. [302], carry out a survey of 1329 scientists across a wide range of sciences and institutions, including, medicine, biology, environmental sciences etc., and concluded that "Barriers to effective data sharing and preservation are deeply rooted in the practices and culture of the research process as well as the researchers themselves". The authors also state that most researchers "agree they are willing to share their data" if "certain conditions are met". Significantly, scientists report "insufficient time and lack of funding" [302] as barriers to scientific data sharing and preservation.

The goal of this case study is to simplify the management of access control to a set of resources, with a particular focus on the ease of sharing in an academic context. In a traditional approach, explicit access control policies for a given resource and a set of users are written, and are then maintained by adding and removing users as, for example, relationships or employment situations change.

The approach advocated in this case study allows for the writing of a policy that adapts to these changes in the environment. In particular, user eligibility changes over time, but these changes are captured automatically. Figure 4.2 illustrates how the GRAFT framework is used to capture recommendation information from two different academic

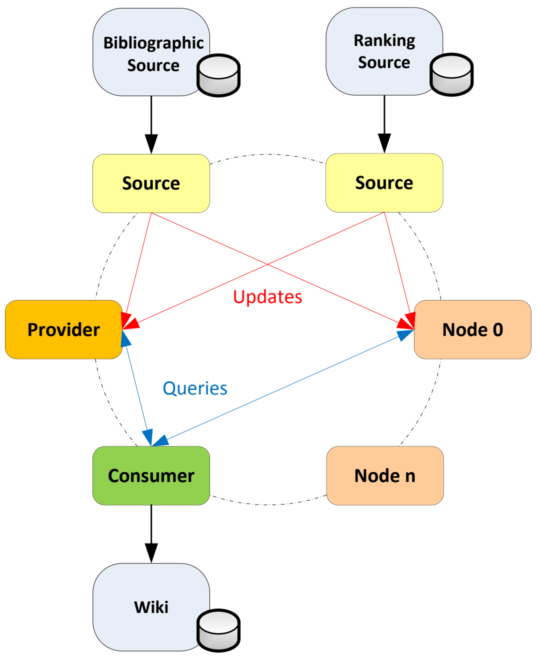

Figure 4.2: GRAFT enabled Wiki. This figure illustrates how the GRAFT framework is used to capture recommendation information from two different academic sources: a bibliographic source and a ranking source.

sources: a bibliographic source and a ranking source. These sources and the information obtained from each is described below:

- **The bibliographic source** provides information about academic papers and their authors. In particular, this source can be used to derive information about co-author relationships. This allows the creation of chains of co-authors, such that it is possible to determine co-authorship relationships separated by several degrees. This is called "degrees of co-authorship", and can be expressed as a number, such that the first degree represents all the authors a target author has worked with, while the second degree is all the authors that the 1st degree authors have worked with, and so on.

- **The ranking source** provides information about the academic standing of authors. An academic rating based on the number of papers and citations for a given author is provided by this source. A metric that is often used in this space is the Hirsch Index (H-index) [142]. The H-index value is computed from the number of publications and citations per publication for an author. It is worth noting that H-index values are not consistent across disciplines [142] and as such the examples given below use only relative values.

The information that each source provides can be used to complement the other, and together they provide a more complete picture of a target author. In this case study, the type of access control made possible by these two sources utilises a two-tier approach, where access to a given wiki page is controlled by the degrees of co-authorship relationship to the page owner, and subsequent editorial rights to the page are controlled by the academic standing of the access seeker. For example, a wiki page created by Alice may have a policy attached that states that her co-authors (1st degree) have access, and that those with a medium academic rating may have the ability to comment, while those with a good academic rating may have edit access. If Bob publishes a paper with Alice, then he automatically gets access to the wiki page, and his academic rating controls the level of editorial access that he has within the page. If Bob is a junior researcher, then he may get read-only access, while a more seasoned researcher gets the ability to comment and those producing influential works are given edit access.

Policy

The policies in the wiki grant access rights to participants based on their academic memberships and standing.

$$x \in S \tag{4.2}$$

Equation 4.2 describes a policy that uses set membership to control access to a resource. If an entity x is in set S, then access will be granted. In this case study, S represents a "degree of co-authorship" to the page owner to which an access seeker must belong in order to be granted access to a particular wiki page.

$$(x \in S) \wedge (r \geq t) \tag{4.3}$$

Building on the previous policy, Equation 4.3 combines both set membership and a threshold check. If the entity x in question belongs to set S, and their academic rating r exceeds threshold t, then access will be granted. In this case study, S represents a "degree of co-authorship" to the page owner to which an access seeker must belong in order to be granted access, while t represents a threshold that controls a particular editorial right the access seeker is able to obtain. The thresholds, and the features made available by reaching each one are specific to each site.

4.2 Composition

Service Oriented Architecture (SOA) is an approach that utilises software services as "fundamental elements for developing applications/solutions" [240]. Services are implemented using standard protocols [347, 66] and can represent any level of functionality, from simple request/response through to complex processes [240].

Web services are an implementation of SOA that utilise XML-based standards [353] to deliver services that are loosely coupled, interoperable [240, 363, 361] and location independent [240]. The wide availability of standardised web services makes it possible to build new functionality by composing multiple web services together [353]. The ability to compose new functionality by combining existing web services from different providers is a key concept in service-oriented computing [240, 308]. However, there is some complexity in the composition of web services, in part due to the large number of offerings available [255].

In this section, GRAFT is utilised in the composition of services. In the first instance, GRAFT provides recommendations to a composing service. These recommendations allow the service to determine the best possible service to select in each instance where there are multiple choices. In the second instance, GRAFT is utilised to provide recommendations for sources of information, and for selecting services from multiple choices.

4.2.1 Travel Planner Service

Consider a "Travel Planner", based on that given in Zeng et al. [353]. In this example a composite travel planning web service is created from the aggregation of a number of specialised services: flight booking, hotel booking, attraction search, driving-time calculation, bike rental and car rental, see Figure 4.3 for an overview.

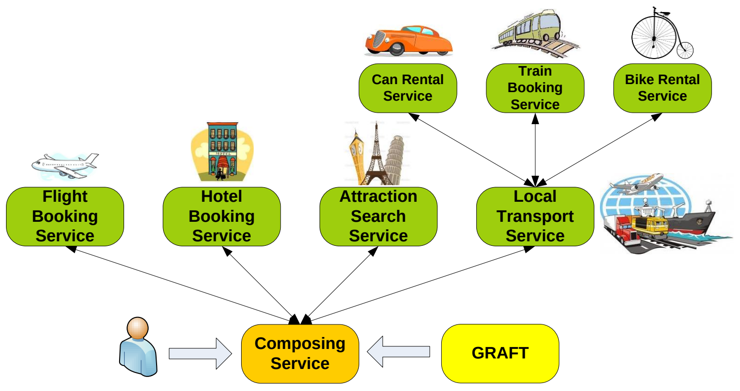

Figure 4.3: An example of Web Services Composition for a Travel Planner.

In a typical Travel Planner interaction, a client submits their travel request and their personal search criteria (e.g. cost vs time, day of travel, etc.) to a selected Composing service, which in turn submits the various aspects of the user's request to the encapsulated services. However, some of these services may have alternatives or substitutes available from different providers. The number of alternatives available for a given service may be large and constantly changing [352]. Therefore, when composing the new service, the question arises: which instances of these specialised services should be used in the new service? Both the client's search criteria and the requirements of the composing service can impact the choice of encapsulated service, in which case, such selections need to be made dynamically. The criteria of the composing service may include: price, reputation and quality of service (QoS) parameters such as availability and reliability [353]. The exact balance of these QoS parameters and the client's criteria are determined by the composing service.

Figure 4.4 illustrates how GRAFT is used as a clearinghouse of recommendation information for use in the ranking of, and reporting on, the components of the composing travel service. The composing service queries a ranking service for the services it needs to fulfil the client's request and supplies its own ranking preferences with the client's criteria as a modifier. The ranking service, in turn, encapsulates a discovery service that returns matching candidate services and then ranks them by obtaining the GRAFT profile for each returned match. The ranked set of profiles are then returned to the composing service which makes the final selection. Multiple ranked services are returned, as potentially some services may be unavailable or semantically incompatible [208]. Once the encapsulated services are selected by the composing service, the client's query is processed and the performance of the encapsulated services are reported back to GRAFT via a reporting service.

A similar sequence of events may occur at different levels, such as, the client initially selecting the travel planner, the ranking service selecting the discovery service, and so on.

4.2.2 Meteorological Workflow

Linked Environments for Atmospheric Discovery (LEAD), Gannon et al. [117], is an ideal scenario in which to consider how a recommendation mechanism such as GRAFT might be integrated into a workflow. To clearly distinguish LEAD from the speculative extension described in this section, it will be referred to as GRAFT Enabled LEAD (GE-LEAD). LEAD is a project that focuses on the capture, storage, analysis and visualisation of atmospheric data, with the goal of increasing the accuracy and timeliness of predictions related to hurricanes and tornadoes. Weather prediction is traditionally broken into four phases: data collection, data assimilation, prediction and generation of visualisations. LEAD's approach to this problem is to introduce flexibility into each of these four phases. In

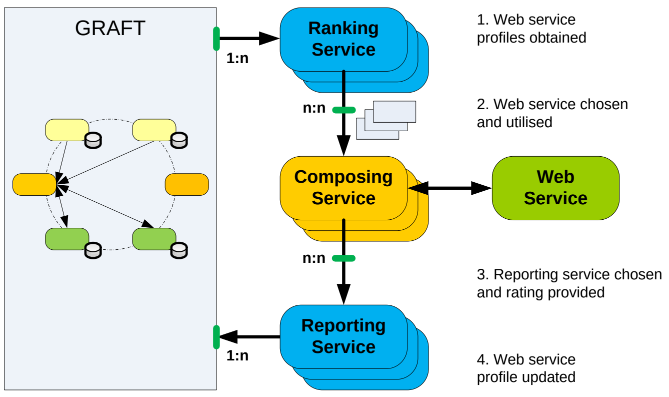
GRAFT
1:n
Ranking Service
n:n
Composing Service
Web Service
n:n
Reporting Service
1:n
1. Web service profiles obtained
2. Web service chosen and utilised
3. Reporting service chosen and rating provided
4. Web service profile updated

Figure 4.4: Web Services Selection; GRAFT is used as a clearinghouse for service recommendations.

particular, LEAD utilises a portal that allows users to create, manage, and monitor workflows as they are executed. The high level relationship between LEAD and GE-LEAD is illustrated in Figure 4.5.

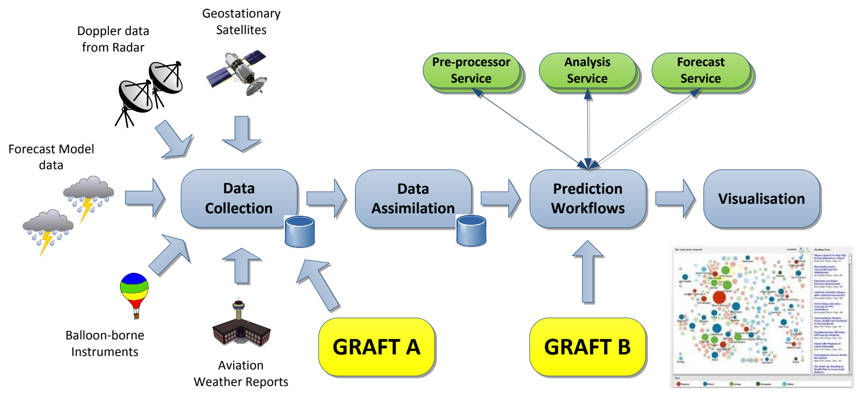

Figure 4.5: A simplified example of Web services composition in the LEAD [117] workflow engine augmented with the decision points at which GRAFT recommendations are applied in GE-LEAD.

LEAD is built using Service Oriented Architecture, with web services organised into a three tier model. The first tier represents basic services such as authentication. The second tier provides data management and workflow execution, while the final tier represents applications "wrapped" as web services so that they can be utilised by LEAD. LEAD employs a Resource Catalogue that describes available services and their parameters. These descriptions are referenced by the workflow composer and allow scientists to compose new workflows based on the service descriptions. A key feature of workflows in LEAD is that they may be modified by external events. In particular, new data may start a workflow, or change the execution of an already in-progress workflow. It is this feature that is extended in GE-LEAD, in particular there are two opportunities within LEAD for the application of GRAFT recommendations. Firstly, in the selection of the workflow (which occurs prior to workflow execution, see GRAFT A, Figure 4.5) and secondly during composition of workflow services (which occurs dynamically during execution, see GRAFT B, Figure 4.5).

Dynamic Workflow Selection

In the case of dynamic workflow selection (GRAFT A), GRAFT provides recommendation information about data sources. For example, when the GE-LEAD workflow engine receives notification of a new data input, it would first query a ranking service for a recommendation on the *expected quality* of the source of the data. Depending on the result, the GE-LEAD workflow engine can process each data source differently. In the case of high quality data, a workflow with no cleansing component can be executed. Where the data is of lower quality, a workflow with a rigorous cleanser may be chosen. In the case of poor quality data sources, GE-LEAD might choose to await a corroborating data source. In the extreme case of a malfunctioning data source, the data may be discarded. The rankings formulated by the ranking service from the GRAFT data can include contextual aspects such as institution, hardware specification, location and QoS (Quality of Service) measures on past performance.

Dynamic Service Selection

In the case of dynamic service selection (GRAFT B), GRAFT provides recommendation information about web services to the workflow engine. A scientist building a workflow can utilise the recommendations in the initial composition of a workflow, or the recommendations can be used dynamically by the workflow engine, to select appropriate (possibly alternative) services at execution time.

Consider a meteorologist at the U.S. National Weather Service (NWS). In creating an initial workflow, she may exhibit a preference for services created or used by immediate colleagues and collaborators, or those from other well known institutions. In addition, she may also only select services for which she has been previously given credentials, and those credentials must still be valid at execution time. In the case of alternative services which may be substituted at runtime when the preferred service is not available, she may well wish to apply a ranking that encodes her implicit trust model and experience. This can be set out in a policy and embedded into the GE-LEAD workflow for reference during execution.

For example, such a set of policies are shown in Equation 4.4, where her first preference is for services from her organisation, followed by those by co-authors and their immediate co-authors (degree of separation of 2) and then by established researchers at top ranked universities (using the Hirsch index [142]). In this sample, m is the member and c is the country. S represents a "degree of co-authorship" to the scientist, while T is the set of top ranked universities. r is the H-index value that must exceed 20.

$$((m = \text{'NWS'}) \wedge (c = \text{'US'})) \vee ((x \in S) \vee ((x \in T) \wedge (r \geq 20))) \qquad (4.4)$$

Once the workflow engine has made the selection, the execution of the workflow continues. The workflow engine later makes a report on the performance of the selected service to the reporting service. This can be done immediately after the service has been used, or towards the end of the workflow as this reporting is not time critical.

4.3 Acquisition

The acquisition of web services components is an important consideration in the utilisation of web services. In particular, as web services become more ubiquitous, access control becomes an important issue that must be addressed [336]. As web services are by their nature distributed, any approach to this problem must be automated and scalable. There are two phases in the construction of a composite web service or workflow. The first is the selection of web services that will make up the final composite entity, while the second is the acquisition of those web services, that is, the demonstration of the right to access and use those services.

Access to a service could be controlled by authentication, however using recommendations allows a service to be more flexible and descriptive as to who may utilise the service. Attributes of the access seeker such as organisation, location and prior behaviour are useful in the aquisition phase.

4.3.1 Travel Planner Service

In the Travel Planner presented in Section 4.2.1, multiple web services are composed into a single web service without reflecting on their actual acquisition. A point to note is that the acquisition decision flows in the opposite direction to the composition decision. In the composition decision, the composer, using recommendations determined which services to use to satisfy a client's request. In the acquisition decision, the selected service determines if the client or the composer is permitted to use the service.

Considering what it would mean to exclude or limit access to services in the travel planner and why it might be useful to limit access, even if the encapsulated service in question was free and public. If the Travel Planner repeatedly sent malformed requests, or overloaded this or other services with impulse loading (which makes it difficult for the service to manage its own resource requirements) then the service might wish to reject the travel planner's requests. In this situation the service itself can request a GRAFT profile for the travel planner to determine if access should be granted. Only once the service has granted access is it possible to say that the service has been acquired by the Travel Planner.

If services are pay-per-use or account based, financial criteria should also be considered, such as credit rating, past payment performance etc. There may also be wider access control considerations, for example, some services may require some form of credential, even if there is no payment. For example, in the travel planner example, access to a flight booking service may only be open to registered travel agents, or made unavailable to direct or co-located competitors.

4.3.2 Meteorological Workflow

In the LEAD workflow scenario described earlier, access to the services that make up a workflow are assumed to be either open, or restricted using OGSA-DAI authentication. As an alternative, it is now worth considering the application of GRAFT malleable access rights [136] in the acquisition phase of the LEAD workflow (applied during GRAFT(b), Figure 4.5). GRAFT malleable access rights permit more flexible and richer access policies to be defined in a GE-LEAD scenario. In addition they are not only easier for a provider of a service (in this case most often a professional scientist, not a software developer) to maintain, as they do not require exhaustive lists of access permissions, but rather encourage a more natural, or perhaps humanistic, set of descriptive access principles to be defined.

Returning to the example of the meteorologist at the National Weather Service. She wishes to use a new forecast service in her workflow, unfortunately the service is only available at Indiana University where the prototype was developed and she is not known to the author. The author of the service, is not yet ready to make it public, as it is critical to his own research and not provisioned for large scale use. At present the meteorologist's only recourse would be to approach the author of the service and request explicit permission to use the service. This would involve considerable overhead in the creation of new credentials and the consequent time delay incurred during this process.

Consider instead, the same situation with GRAFT, in which the author of the service has defined a set of access policies that will permit fellow researchers to experiment with the new forecasting service, while retaining significant control over access. One way to limit access would be for our author to only give access to collaborators and recognised researchers at top institutions, as these are the people that he is most interested in obtaining feedback and data from. The approach taken earlier in this chapter was to utilise degrees of co-authorship or H-index measures to determine the set of collaborators.

4.3.3 Social Cloud

The Social Cloud [68] is another scenario in which GRAFT can be used for service acquisition. The Social Cloud endeavours to utilise Social Networking for the construction of a distributed resource sharing infrastructures. Social networks have seen massive change and growth since the launch of SixDegrees [48], for example, Facebook now has over 1.1 billion users[3], with 50% of social network users checking their pages 3 times a day or more [86].

Boyd [48] provides a number of definitions for social networks, including that social networks allow users to *articulate a list of other users with whom they share a connection*. This feature of social networks is leveraged by Social Clouds to allow the sharing of resources. In particular, the Social Cloud uses existing relationships established in social networks to control sharing of resources. Although controlling access via social constructs, such as "circles", is a trivial task when the number of friends is small, it quickly becomes problematic as the number of friends increases. Indeed, studies have shown that the mean number of friends exceeds 240 people [320].

One approach to this problem is to enhance the friend relationship data that is held in the social network with additional metadata. This approach has been called a socio-technical adapter [64], and can be provided by GRAFT to help enrich the friendship data held in social networks.

The simplest form of a Social Cloud is a photo storage service [69], if this were extended to permit sharing of photos amongst friends, then an access policy in which family and close friends are given access to photos is shown in Equation 4.5. Close friends are defined in this policy as those with a direct friendship relationship (no friends of friends) who have regularly kept in touch (more than 100 interactions). Work colleagues (presumably also excluding the employer) are explicitly excluded. This policy is determined (as in the other examples) by the owner of the photos. In this sample policy, F represents the set of family members, while R represents the "degree of friendship" relationship and W the set of co-workers.

$$(x \in F) \vee ((x \in R) \wedge ((i \geq 100) \wedge (x \notin W))) \tag{4.5}$$

In a Social Compute Cloud [51], the requirements are very different, and indeed more closely align with the policies presented in the LEAD case study in Section 4.2.2. When considering technical services, such as compute, users are more likely to prefer to provide

[3]`https://newsroom.fb.com/Key-Facts` - last accessed November 2013.

resources to colleagues and collaborators. However, a close relationship with the resource requestor is not required, it is sufficient to develop a loose metric for *competence* to avoid misuse.

$$(x \in C) \wedge (x \in S) \tag{4.6}$$

Equation 4.6 presents one such example, from the point of view of a researcher in Computer Science. In this example the Computer Scientist prefers to grant access to fellow computer scientists, whom she feels are more likely to be reliable, but places a limitation on social distance by requiring a maximum degree of separation based on co-authorship (so that social pressure can still be realistically applied in a case of misuse). In this sample, C represents the community, while S represents the "degree of co-authorship".

Chapter 5

Implementation

This chapter discusses the prototype implementation of the GRAFT framework, including implementations of the forum, wiki and workflow case studies presented in Chapter 4. The prototype implements two GRAFT providers (GP), a number of generic nodes, and a series of source and consumer nodes as appropriate for each case study. The nodes in the prototype are spread equally across two hosts. The prototype implements the integrated approach for profile retrieval, allowing recommendation consumers two ways to access profiles.

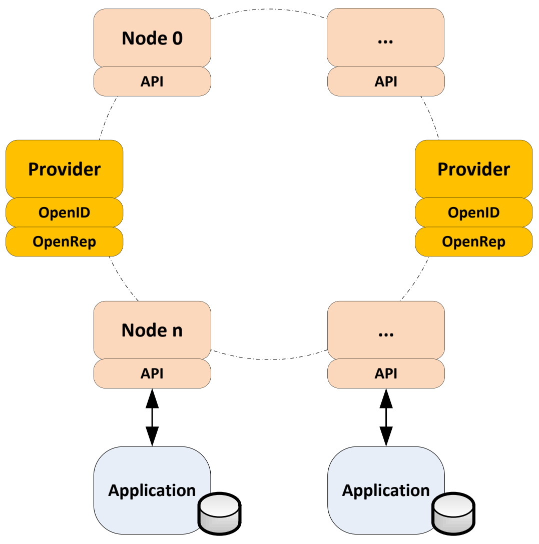

Figure 5.1: GRAFT Prototype Framework. The key components in the prototype, including the providers and the nodes, are shown in this figure.

Figure 5.1 illustrates the key components in the prototype, including the providers and the nodes. The underlying Kademlia network is implemented using the Python Entangled library[1]. This library implements a fully functional Kademlia-based DHT.

- **The providers** are based on SimpleID[2] which implements a complete OpenID provider in PHP. This package has been extended with OpenRep support[3], including the ability to maintain a single profile per registered entity.
- **The nodes** in the network were authored in the assumption that source and consumer applications will not have full GRAFT integration, or that they are written in a language other than Python. To this end, each node implements a TCP API that allows it to communicate with their respective source or consumer applications and query or update profiles. In an ideal scenario, a library would be available for all popular languages, allowing the application to participate in the network directly.

5.1 Forum

A standard feature of many web based forums is the ability for participants to confer "karma" on authors of particularly relevant messages by "up voting" those messages. This karma is summed and the resulting karma score can then be used to obtain additional rights or privileges within the forum. A good example of this approach is the technology forum SlashDot [13], where authors with good karma obtain better default visibility for their messages.

This section discusses the implementation of a consumer forum that utilises GRAFT to implement karma scores. This implementation is in fact a re-implementation of the karma feature found on many forums, and is based on the Web-based forum case study discussed in Section 4.1.1. As in the case study, there is both a source and a consumer forum, with GRAFT acting as the intermediary between the two.

Recommendation information, in the form of karma scores is generated in the source forum as a component of normal use. The source node makes this information available in GRAFT so that it can subsequently be used in the consumer forum. Participants in the consumer forum can use the recommendation information to bootstrap their karma score at the consumer forum when they are newly registered users, or use it to augment their karma score if they are existing users.

Figure 5.2 demonstrates the architecture of this two forum implementation. It should be noted that this figure illustrates only the key information flows. All users are registered at one of the two providers, and use their OpenIDs to authenticate to the source

[1]`http://entangled.sourceforge.net/` - last accessed October 2014.
[2]`http://simpleid.sourceforge.net/` - last accessed October 2014.
[3]OpenRep is further discussed in Section A.2

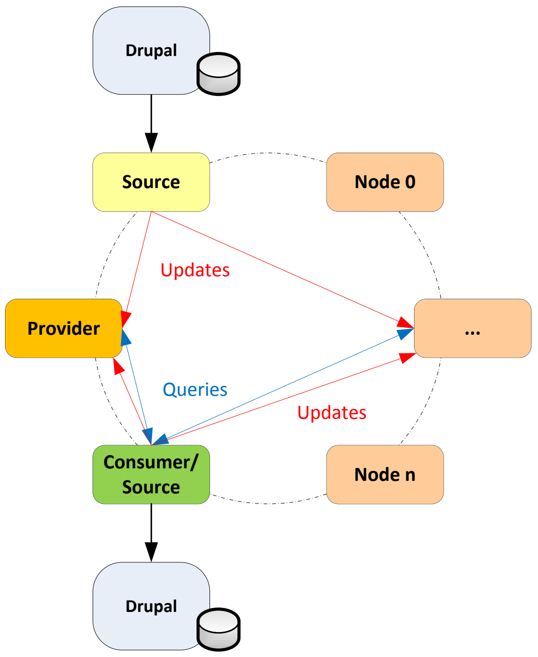

Figure 5.2: GRAFT enabled Drupal Forums. Recommendation information, in the form of karma scores is generated in the source forum as a component of normal use. The source node makes this information available in GRAFT so that it can subsequently be used in the consumer forum.

and consumer forums. Both of the forums are instances of the Drupal[4] framework. This framework was selected for this implementation as it comes with integrated OpenID authentication, and allows for the installation of additional functionality in the form of modules. In this instance the *forum*, *userpoints* and *userpoints karma* modules were also installed. Together, these modules add a highly configurable forum, with user karma functionality, to the base Drupal installation. The two Drupal instances and their respective MySQL databases are installed on separate hosts. The source and consumer nodes are implemented in PHP, as this allows for integration with Drupal, and have read/write access to the underlying Drupal databases.

5.1.1 Source Node

In this implementation the source node implements only limited functionality: the export of recommendation information from the source forum. It is a genuine source-only node, in that it does not use profiles for its own purposes. The source node is integrated directly with the Drupal database, and regularly exports every user's karma score. For every

[4]`http://drupal.org/` - last accessed October 2014.

user, a single SQL JOIN query combines the user's details with their karma score. This information is used to generate a recommendation from itself (that is, the source forum). Using the user's OpenID, the source node is able to fetch the user's profile from GRAFT, amend the profile with the new recommendation, and finally push the profile back to GRAFT[5]. For this prototype implementation, a relatively short delay of 60 minutes was set between update cycles.

5.1.2 Consumer/Source Node

The consumer/source node is so named because in this instance it both consumes recommendation information from another source, and because it is the source for recommendations about its users. The consumer/source node is integrated directly into Drupal. Upon user authentication to the forum, the node fetches a profile for the user from GRAFT. There are two possible scenarios at this point: either this is a returning user, or this is a new user. If the user is a returning user, then the node will be able to find a recommendation from itself, and this is subsequently used to establish the user's karma score. If the user is new to the forum, then the profile will not contain a recommendation with itself as the source, and instead it searches for a recommendation from another source that has the same context as itself. This value can be used as the karma score for the user in the consumer forum if it trusts the source of the recommendation. This trust could be established by examining the profile of the source, although that step has not been implemented for the consumer node.

In using the karma score from the source node, the consumer node may apply a weighting. If the level of trust it places on the source forum is high, then it may put additional weighting on the karma score, and conversely, if the trust is low then a lower weighting is appropriate. In this instance the weighting was set to 0.8 (high level of trust) as trust evaluation was not implemented.

Finally, upon logout, or at a regular interval, the consumer node will generate a recommendation (with itself as the source) that will be placed in the user's profile. This recommendation will be present in the profile the next time the user authenticates to the consumer forum.

5.1.3 Policy

On the consumer forum, policy is enforced using Ruler[6], a stateless rules engine for PHP. The following equations describe two common situations in the forum. In Equation 5.1, read access is restricted to a given sub-forum or thread, based on the user's karma score

[5]A sample profile can be found in Section D.1

[6]`https://github.com/bobthecow/Ruler` - last accessed October 2014.

and administrative status.

$$(rating \leq 10) \vee (\text{administrator} = false) \tag{5.1}$$

In a similar fashion, Equation 5.2 is used to grant the moderator role to users that exceed a karma score of 100, and are not in the "administrator chat" sub-forum.

$$(rating \geq 100) \wedge (forum \neq \text{“administrator chat”}) \tag{5.2}$$

5.2 Wiki

In a survey carried out by Tenopir et al. [302], scientists were "willing to share their data", however a lack of time and funding was an impediement for many. Access control is an important component of data sharing, traditionally requiring the ongoing maintenance of access control policies and user lists. This maintenance can become particularly time-consuming as the number of users grows.

This section discusses the implementation of a wiki that leverages GRAFT to simplify the management of access control. In particular, an access control policy is authored once, with regular updates from GRAFT obviating the need to maintain user lists. This implementation is based on the Wiki first introduced in Section 4.1.2. Similarly to the case study, two academic sources provide the recommendations used to drive access control in the wiki.

Figure 5.3 illustrates the architecture of this implementation. There are two sources of recommendation information, and a single consumer in the form of a wiki. The two sources are the co-authorship relationship and academic ranking sources. The Wiki is an instance of the highly flexible PmWiki[7] application written in PHP. The GRAFT consumer node is integrated directly into the Wiki.

5.2.1 Source Nodes

This implementation uses two recommendation sources: a bibliographic source and a ranking source. Both of these sources are source-only nodes, as the nodes do not consume the profiles, other than in the creation of recommendations that are subsequently placed into the profiles. Both sources are described below.

Bibliographic Source

This recommendation source provides information about academic papers and their authors. In particular, this source is used to generate "degrees of co-authorship" as this

[7] `http://www.pmwiki.org/` - last accessed October 2014.

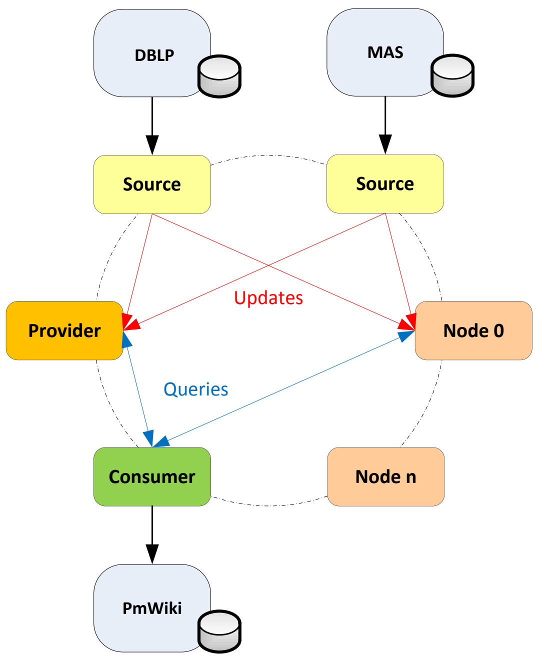

Figure 5.3: GRAFT enabled PmWiki. Two sources of recommendation information feed a single consumer in the form of a wiki. The two sources are the co-authorship relationship and academic ranking sources.

information is not readily fetchable from the Internet.

DBLP[8] was selected for this implementation as its main database is published under the Open Data Commons Attribution License (ODC-BY 1.0)[9], and is available for download in XML format[10]. DBLP provides an online bibliography of a large number of computer science journals and conferences. The information it makes available for each published item includes a list of the authors, which can be used to construct the degrees of co-authorship relationships described in the original case study.

The downloaded bibliographic database was converted from XML and written to a PostgreSQL database. This database provides the source with the ability to query the publications belonging to a given author, and from each of the results extract the co-authors. This provides a first degree relationship. The second degree relationship, co-authors of co-authors, is subsequently extracted by querying each of the co-authors, and so forth. It should be noted that the prototype implementation does not differentiate between two authors with the exact same name.

[8] `http://www.dblp.org/` - last accessed October 2014.
[9] `http://opendatacommons.org/licenses/by/summary/` - last accessed October 2014.
[10] `http://dblp.uni-trier.de/xml/` - last accessed October 2014.

As with other GRAFT sources, this source puts its degrees of co-authorship information directly into the relevant profiles. A key difference in this instance though is the fact that co-authorship information is more verbose than simple integer ratings, as it consists of lists of OpenIDs for each "degree". This information is stored in a hash of arrays in JSON format and compressed in order to bring the size down.

As the OpenIDs for all possible authors were not known, the source created dummy OpenIDs to represent most authors, while key authors were replaced with valid OpenIDs so that they could be resolved to profiles.

Ranking Source

This recommendation source provides ranking information for academic authors. As in the original case study, the Hirsch Index (H-index) metric was used as a rating value. This value cannot be fetched for every potential author because not every author has a profile on an academic website, but also because some search engines block robots. Instead, it is computed from an academic source by examining all the articles, and the number of cites that each article has.

In this implementation, Microsoft Academic Search (MAS)[11] was used as the academic source, because it provides a well-documented and usable API to registered users. The API returns results to the requestor in either JSON or SOAP formats. JSON was used in this instance as this is consistent with the approach taken in other parts of the implementation.

A random selection of authors was used to "seed" the ranking source. For each of the authors, the source needs to know the author's name, and a mapping to their OpenID. The source queries MAS for their publications, with the results sorted by citation count. As each result page is limited to 100 results, pages are fetched from MAS on a page-by-page basis until there are no further results. The H-index value is then computed. As with other sources, the profile for each author is fetched, updated and pushed back into the network. There is an underlying assumption in this implementation that H-index values are best computed non-interactively. This needs to be experimentally confirmed.

5.2.2 Consumer Node

The consumer node in this implementation is a genuine consumer-only node, as the wiki does not maintain its own karma or rating scores. The consumer node is integrated into the PmWiki application as a separate module that is included for every page load. Immediately after login, the user's profile is fetched, analysed and the ranking score stored with other session variables such as username in the session cookie. The degrees of co-

[11]`http://academic.research.microsoft.com/` - last accessed October 2014.

authorship information (in JSON format) was deemed too large to store this way and was written to disk. In order to speed up testing, all degree values that were requested during policy evaluation were cached. This ensured that later requests using the same two authors were faster in second and subsequent runs.

Access to each wiki page is controlled by two-tier policy that first examines the co-authorship degree between the page creator and the current user. If the two users are within the number of degrees specified by the policy, then the second tier is used to compute the level of access the user may have to the page. In this case study, users could obtain read, write and moderator access to a page based on their H-index. Similarly to Stackoverflow, the levels were set to 1 (read), 20 (write), and 50 (moderator) respectively.

5.2.3 Policy

On the consumer wiki, policy is again enforced using Ruler. The following equations describe two policies that can be used to limit access to a wiki page based on both the academic standing of an individual and their degree of co-authorship to the page creator. In Equation 5.3, read access is granted to a page based on the user being within three degrees of co-authorship to the page creator, and having a single cited work.

$$(degree \geq 0 \wedge degree \leq 3) \wedge hindex \geq 1 \tag{5.3}$$

Equation 5.4 is used to grant write access to a wiki page. This policy is more restrictive than the previous policy, in that the set of individuals has been limited to two degrees of co-authorship. In addition, the H-index requirement has been raised to a significant number of published and cited works.

$$(degree \geq 0 \wedge degree \leq 2) \wedge hindex \geq 20 \tag{5.4}$$

5.3 Workflow

In the workflow case study presented in Section 4.2.2, an example of dynamic service selection was presented. Dynamic service selection allows a workflow author to build workflows where web services are selected at runtime, rather than being encoded in the workflow. This implementation demonstrates a standard workflow engine that has been augmented with the ability to dynamically select a web service based on their recommendation information.

As shown in Figure 5.4, GRAFT provides recommendation information, via a ranking service, to the Anduril workflow framework. The Anduril[12] workflow framework [236]

[12]`http://www.anduril.org/` - last accessed October 2014.

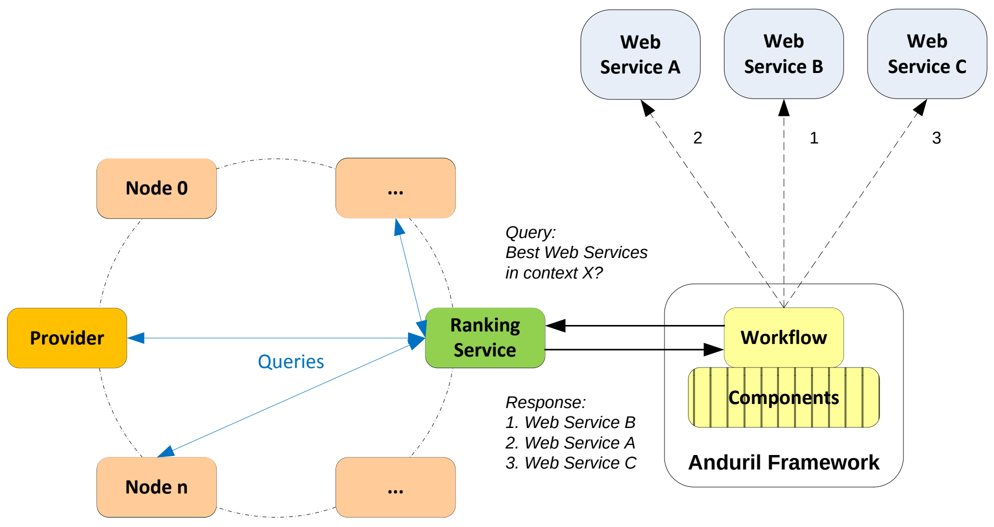

Figure 5.4: GRAFT enabled Anduril Workflow. GRAFT provides recommendation information, via a ranking service, to the Anduril workflow framework.

was developed to provide a component-based approach to scientific workflows, with a particular focus on those in biomedical research.

Within Anduril, each workflow consists of a number of commands, and each command is defined by an Anduril component. Some components are provided by the core engine, however additional components can be authored in a variety of languages. All of these components are loaded by Anduril upon start-up, and utilised by the workflow engine as required.

The implementation in this section delivers two additional components for Anduril, and a ranking service for GRAFT. The two new components for Anduril provide it with the ability to call a web service, and the ability to call the GRAFT ranking service.

5.3.1 Anduril Components

The first new component, called *CallService*, takes a list of URLs and a data file as parameters. The list of URLs are tried in order, using HTTP, until one succeeds within a pre-defined number of retries. The contents of the data file, if it exists, are passed along with the call, and the results from the web service are returned to the workflow engine.

The second new component *RankService* takes only a context as a parameter. It calls the GRAFT ranking service with the context and returns a list of matching web services, in the form of a URL list, to the workflow engine. This list can subsequently be passed to the CallService component. For ease of implementation this component only knows the location of a single ranking service.

5.3.2 Ranking Service

The ranking service is a prototype web service that provides a ranked list of services that match a requested context. Upon request, the ranking service returns an ordered list of URLs that match the given context. Each URL is also the OpenID of the web service.

In a real-world implementation, the ranking service would provide ranked lists of services to requestors, and accept feedback on the performance of services it had previously recommended. Accepting feedback would allow it to track the performance of services over time, allowing it to adjust its internal rankings as the situation changes.

In this prototype implementation, the ranking service has a number of deficiencies. The first is that the ranking service has no ability to accept feedback on its previous recommendations. Secondly, the ranking service is aware of only a fixed number of contexts. Each context is represented by a string that designates the name of the context. For each context, the ranking service maintains a list of OpenIDs, sorted according to their rating. Finally, the act of fetching and ranking the profiles only occurs once, during initiation of the service. The profiles are ranked by the ranking service recommendation found in each profile.

5.3.3 Anduril Workflow

As shown in Figure 5.5, the workflow first calls the RankService component in order to get a ranked list of services matching the given context. The workflow then attempts to call the best matching service. As noted above, a fall-back mechanism in the CallService component ensures that failures are handled gracefully, by trying the next best match when the first fails[13].

[13]Sample Anduril workflows are provided in Section D.3.

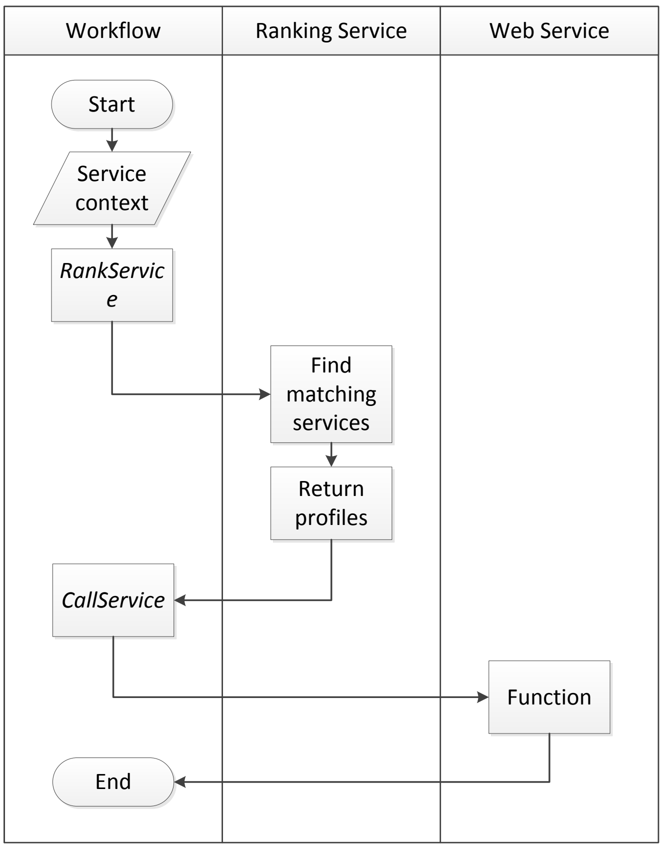

Figure 5.5: The workflow first calls the RankService component in order to get a ranked list of services matching the given context. The workflow then attempts to call the best matching service.

Chapter 6

Experimental Results and Evaluation

This chapter describes the experiments that were conducted on the GRAFT framework prototype which was introduced in Chapter 5. There are three important aspects of the framework which require experimental validation.

Firstly, sources of recommendation information are critical to the architecture as without these, recommendation consumers do not have any information to consider when making decisions. Section 6.2[1] and Section 6.3[1] describe experiments conducted for the "degrees of co-authorship" source, which show it is a feasible source for non-interactive profile updates, while Section 6.4[1] discusses an experiment that analyses Hirsch Index calculations using Microsoft Academic Search (MAS). This experiment also shows the feasibility of this source for non-interactive profile updates.

Secondly, recommendation consumers must be able to efficiently utilise the architecture to provide services, but without incurring significant overheads. Section 6.5 discusses an experiment with the prototype workflow implementation that examines the performance of the ranking service implemented in Chapter 5. This experiment shows that the ranking service adds only a small overhead when choosing web services in a dynamic workflow environment.

Lastly, the architecture must be able to scale and efficiently provide profiles to consumers, even when under churn and in the presence of malicious peers. A key aspect of this includes the ability for consumers to be able to verify the profiles they have received. Section 6.6 builds a simulation of the ranking service that is scaled up to thousands of nodes. This experiment validates the distributed architecture at a large scale, and shows its usefulness in time-critical applications up to approximately 400,000 nodes.

The application and execution of policies were considered a component of the implementation discussed in Chapter 5, and are not further addressed here.

[1]The work in this section is largely taken from the published paper F. Hendrikx, K. Bubendorfer, Malleable access rights to establish and enable scientific collaboration, in: 9th International Conference on eSciece, IEEE, Beijing, China, 2013

6.1 Experimental Testbed

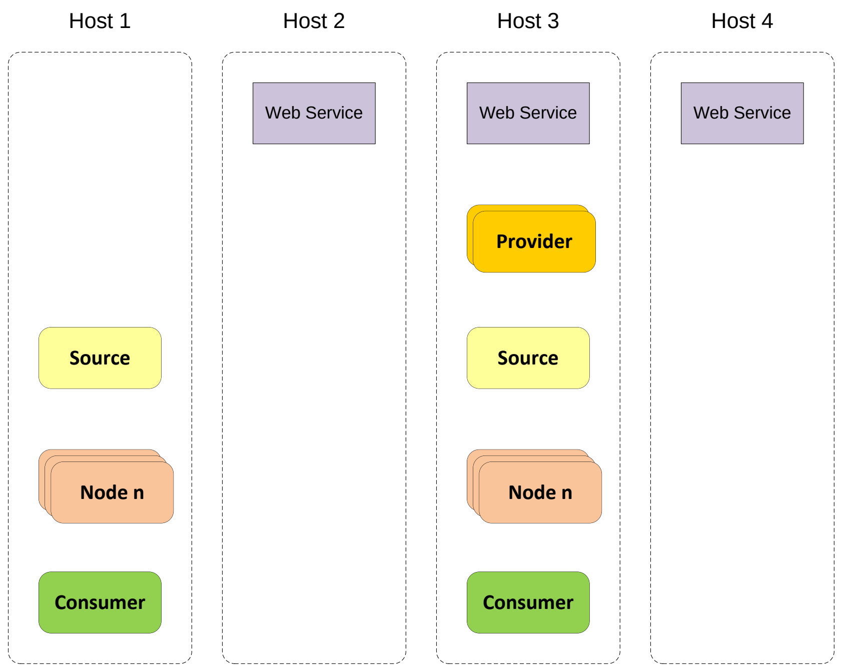

Figure 6.1: Evaluation Testbed. The testbed comprises four dedicated hosts.

All of the experiments presented in this chapter, except for the simulation work, were conducted on a single testbed that is spread over four dedicated hosts. Figure 6.1 is a high-level view of the experimental architecture. Each host is further described below.

- **Host 1:** AMD Athlon 64 3200+, 4 GB RAM, Debian/GNU Linux 7.4
- **Host 2:** Intel Core i7 2.0 GHz, 8 GB RAM, Windows 7
- **Host 3:** Intel Core 2 Duo 3.0 GHz, 4 GB RAM, Debian/GNU Linux 7.4
- **Host 4:** Intel Xeon E5640 2.67 GHz, 8 GB RAM, Debian/GNU Linux 7.4

Hosts 1 and 2 reside on the same local LAN, while Host 3 is within the same top level country domain, with an average latency of 50ms from hosts 1 and 2. Finally, host 4 is approximately 18,500km away and has an average latency of about 350ms from hosts 1 and 2. All four of the hosts were dedicated to the experiments, with non-essential services shutdown before proceeding with any measurements.

Each experiment was conducted on between 1 and 4 of the hosts in the testbed, depending on the exact nature and requirements of the experiment. The experiments described in Section 6.2, Section 6.3 and Section 6.4 utilised host 3 (as this was the first host acquired for the evaluation), while those in Section 6.5 utilised all of the hosts.

All performance measurements are taken from either the perspective of the consumer or the service provider, depending on the requirements of the experiment. Consumers log all actions taken, including when requests are made and when responses are received. Similarly, service providers log incoming requests and when responses to those requests are sent out again.

6.1.1 Prototype Limitations

The prototype discussed in Chapter 5 had a number of limitations that could affect the experimental results presented in this chapter.

In particular, the prototype was a "proof of concept" that used a distributed network implemented in one language, while the core logic was written in another. Communication between these components was achieved through TCP sockets, potentially adding unnecessary delays. Further, the extensive use of interpreted languages in the implementation of GRAFT (such as PHP and BASH) may cause additional slowness when compared to faster C or C++ implementations.

The slower execution caused by the choice of languages is likely to have had an effect on the results presented in this chapter. In particular, the evaluations discussed in Section 6.2: Degrees of Co-authorship Calculation, Section 6.4: Hirsch Index Calculation and Section 6.5: Workflow are likely to be slower than they might have been if they have been implemented in another language. As such, the results presented should be considered worst-case scenario.

6.2 Degrees of Co-authorship Calculation

In Section 5.2, a bibliographic source and a ranking source were implemented for the Wiki case-study. The bibliographic source was used to generate degrees of co-authorship information from a DBLP database, and allowed a page creator to give access to his or her co-authors, or co-authors of co-authors.

6.2.1 Design

On his website, usability expert Jakob Nielsen talks about website responsiveness [226]. In particular, Nielsen notes that humans do not perform well when they have to wait for a result, as this taxes our attention and short term memory. Humans also like to feel in control, and a long wait leaves us feeling at the mercy of a computer. Nielsen proposes three response time-limits for web usability (June 2010). The first limit is 0.1 seconds, where a response feels instantaneous, and as a result, the user experiences a sense of direct manipulation. The second limit is 1 second, where the user's thoughts remain uninterrupted, and hence they retain a feeling of control, despite the slight delay. Users will tolerate longer delays; however, at 10 seconds (the final limit) the user's attention span limit has been reached. Users experiencing this level of delay will often leave a website immediately.

This experiment considers the time required to fetch and compute increasing degrees of co-authorship. In particular, this experiment was intended to be run until the time taken to compute any single degree was clearly no longer suitable for an interactive application. According to Nielsen, this would limit the time to 10 seconds at most.

Degrees of co-authorship were calculated using a local copy of the DBLP database running on PostgreSQL. A sample of twenty computer science authors was selected using the number of publications they each had, ensuring that the sample included authors with both small and large numbers of publications. For each author, their actual publications were determined, and from that their set of unique co-authors. The set of co-authors were then used to determine their publications and co-authors, and so on for each additional degree.

6.2.2 Results

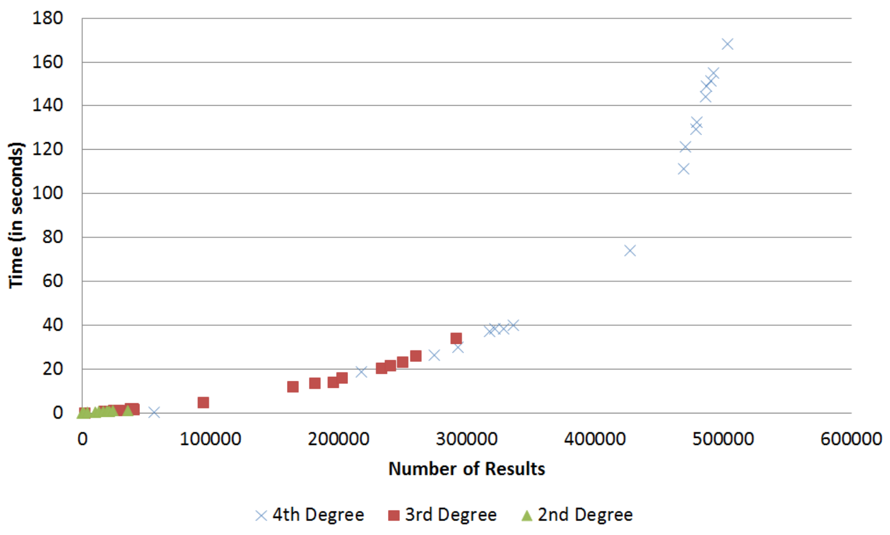

Figure 6.2: Degrees of Co-authorship vs Time in seconds. Only the 2nd, 3rd and 4th degree co-authorship results are plotted in this graph, because the 1st degree co-authorship results are measured in thousandths of a second and are not visible at this scale. Error bars for the 95% confidence interval are also plotted, but are not visible due to their small size.

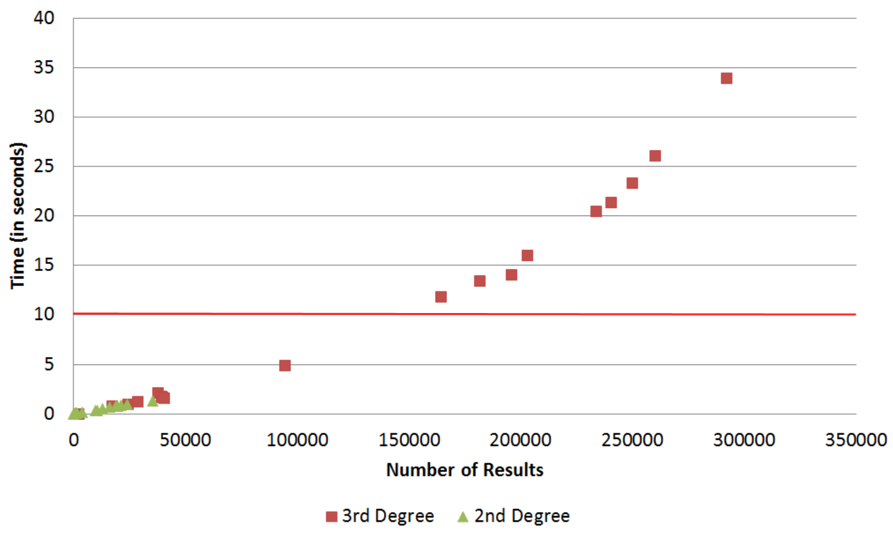

Figure 6.3: Degrees of Co-authorship vs Time in seconds, containing only the 2nd and 3rd degree results. The graph shows that the time taken to compute the 1st and 2nd degrees fall well within Nielsen's 10 second limit (shown as a horizontal red line).

The original implementation was written in PHP, but the results were heavily skewed by memory allocation artefacts. The final implementation was therefore written in Perl. The overall results can be seen in Figure 6.2. Only the 2nd, 3rd and 4th degree co-authorship results are plotted in this graph, because the 1st degree co-authorship results are measured in thousandths of a second and are not visible at this scale. Error bars for the 95% confidence interval are also plotted, but are not visible due to their small size.

A version of the same graph, but showing only the 2nd and 3rd degree results can be seen in Figure 6.3. The graph shows that the time taken to compute the 1st and 2nd degrees fall well within Nielsen's 10 second limit (shown as a horizontal red line on the graph). However, the 3rd and fourth degrees are clearly outside this limit. Based on these results, any policies requiring more than the 2nd "degree of co-authorship" should consider using more powerful hardware, or pre-calculating the results and caching them in the author's profile.

6.3 Degrees of Co-authorship Storage

Given the results in Figure 6.2, a sensible conclusion is that the degrees of co-authorship should be pre-computed and stored for each individual author, rather than generated on demand. As co-author relationships will only change slowly over time, this should constitute a reasonable approach.

6.3.1 Design

This experiment considers the storage space required to store complete co-author relationships for increasing degrees of co-authorship. The goal of this experiment was to be able to store entire degrees for an author, without making the profile larger than 3 MB. This value was selected because this is a size that can be easily transferred over a 100 Mbit link in under a second. This bandwidth is typical for a LAN.

The degrees of co-authorship computation described previously was re-executed, and the information obtained was stored in a profile. This experiment ignores the fact that the profile could contain other information, as the co-author information would most likely make up the bulk of the profile.

6.3.2 Results

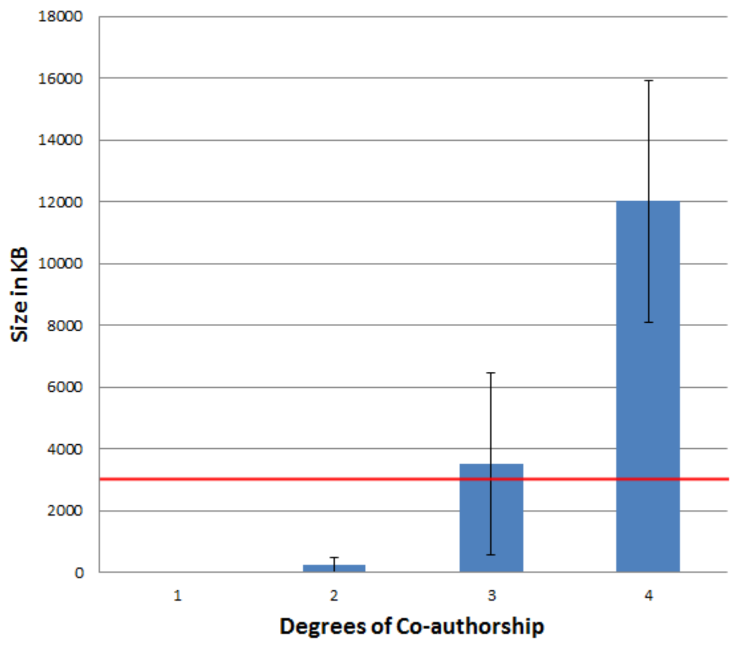

Figure 6.4: Profile sizes with degrees of co-authorship stored. Profile sizes increase as the degree increases. Only the 2nd, 3rd and 4th degree co-authorship results are plotted in this graph because the 1st degree co-authorship results are measured in tens of kilobytes and are not visible at this scale. Error bars for the 95% confidence interval are also plotted.

Figure 6.4 shows how profile sizes increase as the degree increases. Only the 2nd, 3rd and 4th degree co-authorship results are plotted in this graph because the 1st degree co-authorship results are measured in tens of kilobytes and are not visible at this scale. Error bars for the 95% confidence interval are also plotted, which show a large variance in the profile sizes for each degree. Profile sizes exceed 3 MB at the third "degree of co-authorship", meaning that only the 1st and 2nd degree could be cached before becoming too large. Given that the 1st and 2nd degree can be computed with ease, this result is not useful.

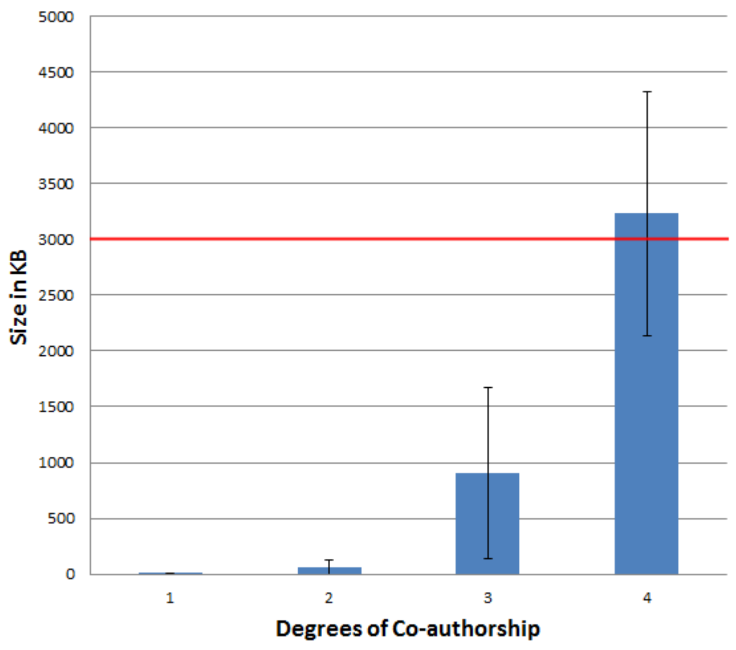

Figure 6.5: Profile sizes with degrees of co-authorship stored compressed. Profile sizes increase as the degree increases. However, in this instance, profile data has been compressed using gzip.

Compressing the degrees of co-authorship was considered next. As with the previous figure, Figure 6.5 shows how profile sizes increase as the degree increases. However, in this instance, profile data has been compressed using gzip. These results show that pre-computing and caching up to the 3rd degree in a profile is feasible, if the degrees of co-authorship information is first compressed. The 4th degree could also be considered feasible for storage in a profile, but only if the 3 MB limit was raised to 4.5 MB.

6.4 Hirsch Index Calculation

In Section 5.2, a bibliographic source and a ranking source were implemented for the Wiki case-study. The ranking source provided academic ranking information for individuals. In particular, this source generated Hirsch Index (H-index) [142] values using Microsoft Academic Search (MAS). This section examines the performance of the ranking source, in order to evaluate its usefulness in on-demand applications.

6.4.1 Design

This experiment considers the average time taken to fetch a page from MAS, as this determines the time to calculate the H-index value for a given author. The implementation discussed in Chapter 5 assumed that H-index values would be computed and stored in a profile, but it is useful to consider how quickly a H-index value could be computed for a given author, and if it falls within the 10 second upper limit given by Nielsen.

As discussed previously in Section 5.2, H-index values were calculated by searching for an author using MAS and downloading each result page. The JSON[2] interface to MAS was utilised because it is consistent with the RESTful approach used in the remainder of this work.

A simple program that requests only a single result for an author search was written, effectively giving a simple "ping" for the MAS API. This program was run every 5 minutes using the CRON scheduler, and fed the name of a random author from the sample list of 20 that was selected for the previous experiment.

6.4.2 Results

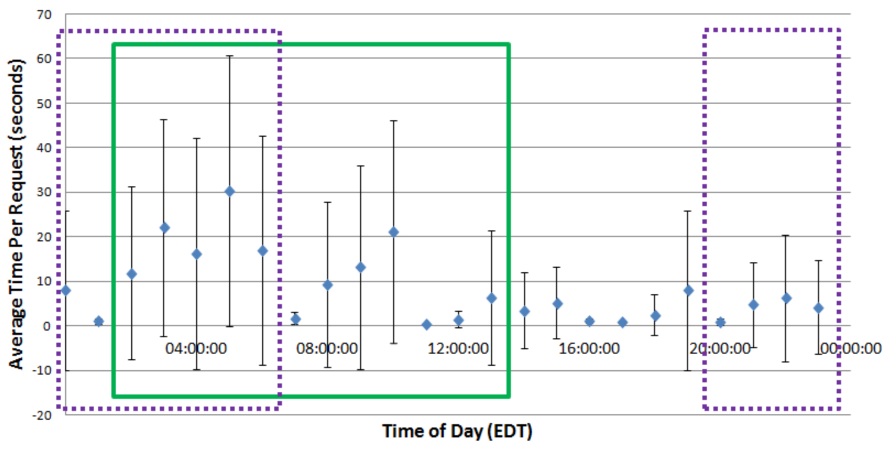

Figure 6.6: MAS Average Page Request Time for 28th April 2013. The average page request time was 8.20 seconds (2 dp), with a standard deviation of 17.20 seconds (2 dp). The error bars show the 95% confidence interval. All times shown are recorded in EDT (Eastern Daylight Time). The dashed purple line shows the Chinese working day (0730 CST to 1830 CST), while the unbroken green line shows the European (including the United Kingdom) working day (0730 CEST to 1830 BST). The greatest variance in the response time occurs when the Chinese and European working days overlap.

[2]JavaScript Object Notation, a human readable format used to transfer key/value pairs between a client and server.

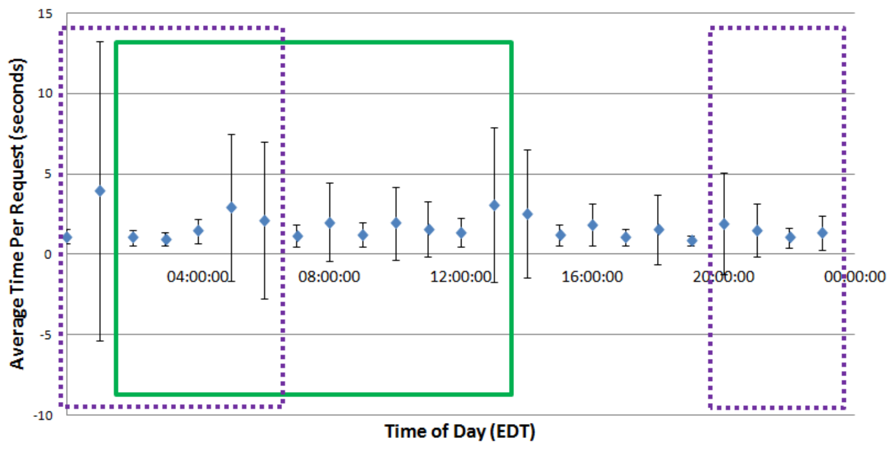

Figure 6.7: MAS Average Page Request Time for 30th April 2013. The average page request time was 1.69 seconds (2 dp), with a standard deviation of 2.93 seconds (2 dp). The error bars show the 95% confidence interval.

Figure 6.6 shows the average time for page requests to the MAS API for Sunday 28th April 2013. All requests were grouped into hourly results and plotted against the average time taken to request and retrieve the result from the MAS API. The error bars show the 95% confidence interval. All times shown are recorded in EDT (Eastern Daylight Time). The dashed purple line shows the Chinese working day (0730 CST to 1830 CST), while the unbroken green line shows the European (including the United Kingdom) working day (0730 CEST to 1830 BST). The greatest variance in the response time occurs when the Chinese and European working days overlap.

For this graph, the average page request time was 8.20 seconds (2 dp), with a standard deviation of 17.20 seconds (2 dp). At 2 standard deviations from the mean, this gives a 42.6 second (2 dp) page response time. This time is clearly outside of Nielsen's limits, especially when considering that some authors have multiple result pages.

Figure 6.7 shows the average time for page requests to the MAS API for Tuesday 30th April 2013. The average page request time was 1.69 seconds (2 dp), with a standard deviation of 2.93 seconds (2 dp). At 2 standard deviations from the mean, this gives a 7.55 second (2 dp) page response time. This is within Nielsen's limits, but once again, would fall outside of the limits in situations where multiple result pages would need to be fetched. This graph exhibits much less variance than the previous graph. One possible reason for this difference could be explained by the timing of the United States spring exams (approximately from 29th April 2013 through to 10th May 2013).

The average single page request time for both days is large, considering the sub-second times normally experienced when fetching a page from a web server. Based on these results, it would be impractical to calculate H-index values on-demand when using

MAS.

These numbers also provide an average upper bound (or worst-case scenario) for calculating a H-index value when using MAS. Given that MAS only allows a maximum of 10 result pages per search, the upper bound on any search can be considered to be $10 \times 42.6 = 426$ seconds (with a 95% confidence). As noted previously, this makes interactive use of this interface impractical, and strongly favours caching this information in profiles.

6.5 Workflow

A prototype workflow implementation using a ranking service was described in Section 5.3. This prototype implementation utilises the Anduril workflow engine. It was installed into the testbed, and the performance was measured in order to gain an understanding of real-world execution times, with a particular focus on the ranking service.

6.5.1 Design

The goal of this experiment was to understand the impact of the ranking service on overall performance of a workflow. Introducing a ranking service for dynamically determining an appropriate web service during workflow execution is only useful if it does not greatly impact the overall time. Understanding the overheads incurred by using a ranking service is crucial for a developer considering the cost-benefit trade off in using the proposed framework, and is especially important if an interactive user is waiting for the results of the workflow.

Three different Anduril workflows were measured in this experiment: base, call and rank. Base represents the simplest "no-op" Anduril workflow and gives a base-case for workflow execution times. Call uses the "CallService" component to call a web service, while rank calls the GRAFT ranking service, via "RankService", and then calls the highest ranked web service. Each of the three candidate web services is an "echo" service, that responds with a copy of the input. 4 KB of data was fed to the web service in each instance.

Each workflow was executed 41 times, with the initial result discarded to ensure that the effects of caching by the operating system were treated equally across all experiments. This is important when measuring results from Anudril because it is file-based and, as such, after the first execution, it has been cached. Measurements were taken with both local LAN-based and Internet-based web services.

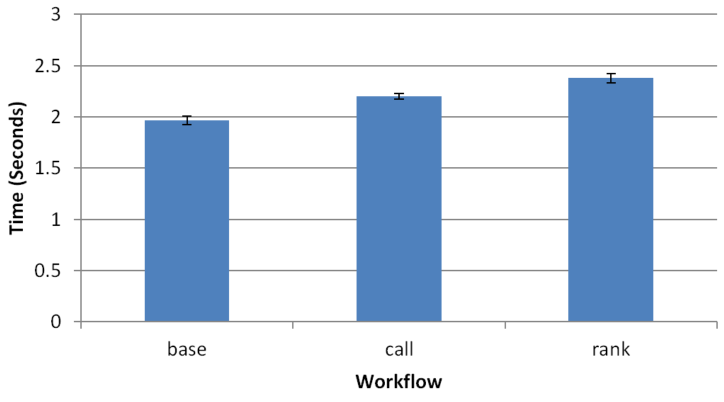

Figure 6.8: Anduril Workflow Execution times for LAN-based web services

6.5.2 Results

Figure 6.8 shows the execution times for the three different workflows using the LAN-based web services. The workflows and ranking service were on host 1, while the web services called by the workflows were all on host 2 (on the local LAN). The error bars show the 95% confidence interval. The overhead of calling a web service is approximately 0.25 seconds, while calling both the ranking service and the highest ranked web service is approximately 0.5 seconds.

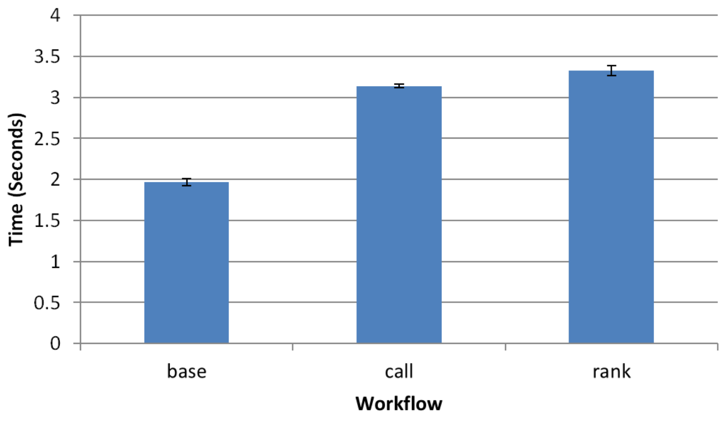

Figure 6.9: Anduril Workflow Execution times for Internet-based web services

Although the overhead of calling the ranking service should not change if it is called for a local or remote service, this is worth verifying experimentally to identify any errors in design or implementation. Figure 6.9 shows the same three Anduril workflows, but with all of the web services on host 3. The overhead of calling a web service from Anduril when calling the Internet-based version is approximately a second. Similarly, calling the ranking service and the highest ranked web service adds just over a second, with the majority of that time taken by the call to the highest ranked web service.

In both the local LAN and Internet versions, the average overhead of calling the ranking service is approximately 0.25 seconds. This was expected because the ranking service resides on host 1, and therefore has a relatively low latency overhead.

6.6 Workflow Simulation

OverSim [40] is a peer-to-peer (P2P) simulator written in C++, which builds on the OMNeT++[3] simulation environment. OMNeT++ is a discrete simulation environment built to simulate communications networks. Modules written in C++ are tied together using a high-level language called NED, allowing the re-use of the components as required. OverSim[4] builds on this infrastructure to deliver an environment that allows for the simulation of peer-to-peer networks, particularly overlay protocols such as Kademlia.

An initial simulation was built using the SimGrid [58] simulation software. However, the results from the simulations were inconsistent and, after careful consideration, it was decided to re-implement the simulation using OverSim. A description of the SimGrid simulation work and the inconsistencies that were found is available in Section C.1.

6.6.1 Design and Testbed

There were a number of goals for these experiments. Firstly, to measure the effect of churn and malicious peers on the architecture's ability to provide profiles to consumers. Secondly, to ensure that consumers can verify the profiles they have received, even with an increasing number of malicious peers. Finally, to consider the scalability of the architecture.

The simulation takes elements from the experiment described in Section 6.5, but does not replicate it entirely. In particular, there is no ranking service, as this would hide the effect of churn and malicious peers from the results.

Figure 6.10 illustrates the key components of the simulation testbed. A composer peer queries a number of peers in the network to obtain profiles they have stored. When it has received the profiles, it can then verify them using threshold voting, and decide which service it is going to utilise. This simulation does not incorporate a separate ranking service, but instead implements the same functionality in the composer peer itself.

The existing Kademlia overlay that comes bundled with OverSim was utilised, with the addition of two new custom node types, both written in C++. The first of these is a GRAFT composer peer that attempts to fetch in parallel the profiles for the services in which it is interested, verify them and then select one service. The second of these new

[3]`http://www.omnetpp.org/` - last accessed October 2014
[4]`http://www.oversim.org/` - last accessed October 2014

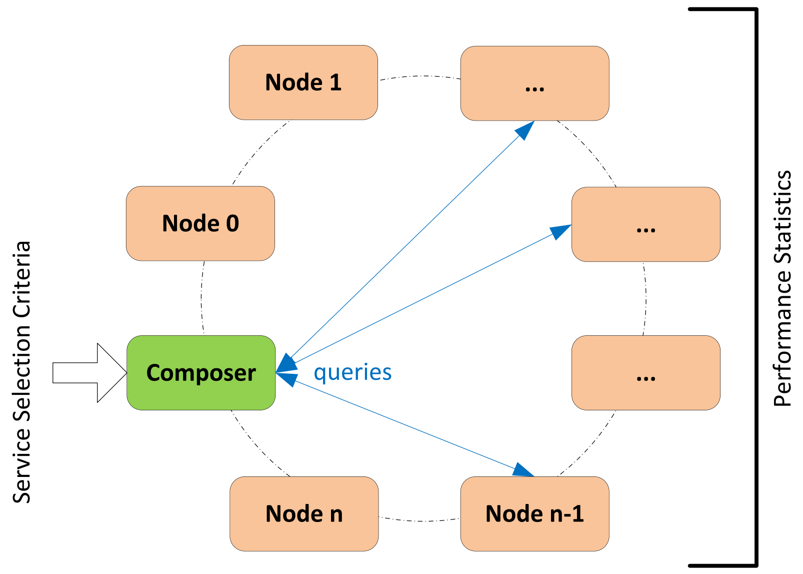

Figure 6.10: Workflow simulation testbed. A composer peer queries a number of peers in the network to obtain profiles they have stored. When it has received the profiles, it can then verify them using threshold voting, and decide which service it is going to utilise.

nodes represents a generic peer node that waits for incoming queries and replies with the appropriate dummy profile. For these experiments, the dummy profiles were set to be 4KB in size. As with the implementation, there was no Kademlia replication between nodes.

All simulations using OverSim were run on an Intel Xeon E5-1650 v2 3.50 GHz, 64 GB RAM, running Arch Linux 2014.02.01.

6.6.2 Parameters

All experiments were run for 9000 seconds simulated time, and were repeated 40 times. The node creation interval time was set to 0.01 seconds, ensuring that all of the nodes required for an experiment were running within 100 seconds, and stable within 1000 seconds simulated time. OverSim provides a feature that allows the parameterisation of key variables, including the seed set used for each experiment. A series of experiments can thus be entirely automated and repeated, with all variables that were used for a particular experiment recorded in the log file[5].

Depending on the experiment, the total number of peers n in the network were varied between 1000 and 9000 peers, in increments of 1000. 9000 peers were selected as an upper

[5]A sample OverSim configuration file that was used in some of the experiments is provided in Section D.4

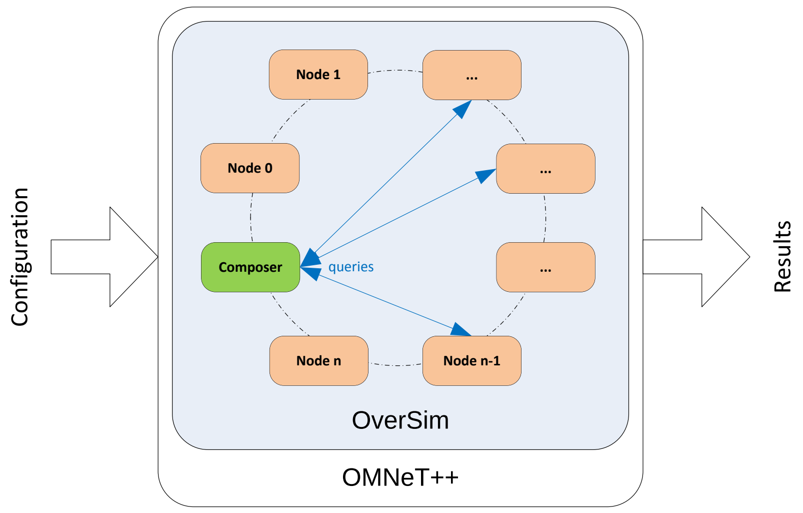

Figure 6.11: Workflow simulation testbed within OverSim. OverSim provides the environment in which the simulation testbed is executed.

limit to the network size, as this meant that a single simulation could run within a few hours. The number of services s in which the composer node was interested was set to either 3 or 5. These were selected as they represent good "multiple choice" situations.

In a similar way, the number of peers p that were queried for profiles was always set to 5, 7, 9 or 11. Having an odd number of peers means that in perfect situations (that is, those with a 100% response rate) the number of responses is always over, or under, exactly half. This ensures that a deadlock situation is avoided when performing a threshold vote, as the vote either succeeds or fails.

The total queries q issued in an experiment is always $p \times s$. A threshold is implemented so that although p peers are queried, the lookup of a service profile is considered complete when a threshold t profile query responses are in agreement. A larger threshold will give more assurance because a greater number of profiles must be in agreement. However, this will also take longer, as more profiles must be received by the composer node.

Two different threshold t proportions were used during the experiments. The first, shown in Equation 6.1 represents the "half" threshold. This threshold succeeds when just over half of the responses are in agreement, and is termed $^1/_2$ *threshold* in the following sections. The second, shown in Equation 6.2 represents the "two-thirds" threshold. This threshold succeeds when just over two-thirds of the responses are in agreement. This is termed $^2/_3$ *threshold* in the following sections.

$$t = \left\lfloor \frac{q+1}{2} \right\rfloor \tag{6.1}$$

$$t = \left\lfloor \frac{q+1}{3} \times 2 \right\rfloor \qquad (6.2)$$

6.6.3 Simulation

New nodes are added to the Kademlia network by OverSim until all the required nodes are present. The composer peer waits for 8000 seconds in simulated time, and then issues profile queries to p peers for each service s in which it is interested, resulting in a total of q profile queries being sent. These queries are sent in parallel, with the composer peer recording the exact time at which each query was sent, and when a matching response was received. The number of services s and peers p is varied for each experiment. The peers p for each run are randomly selected from the available peers before the simulation. The simulation finishes at 9000 seconds.

6.6.4 Presentation

It should be noted that the tables and graphs shown in the following sections often only correspond to the smallest and largest combinations of s and p. Information generated from other variations is often utilised and described, but not always shown. Showing all of the information generated by the simulations was deemed excessive.

6.7 Simulation: Base Case

This section presents the results from the base case simulations. These simulations do not implement any churn or malicious peers and simulate a perfect state. These simulations are intended to provide a base case for later comparisons. In particular, the base case examines three key metrics. The first is the single query response time. The second is the inter-arrival time for responses, while the final is the total time taken to reach a threshold.

6.7.1 Single Response Time

The single response time experiment records the average time taken to send a query for a profile and receive a response. All query response times measured during the simulations were used to compile the graphs shown here.

Figure 6.12 and Figure 6.13 show the average time taken for a single query and response for 3 and 5 services. The variables p (peers queried per service) and n (total number of peers) are plotted against the time taken to receive a response. These graphs both have an average single query and response time of 0.62 seconds (2 dp) for 5 peers, rising to 0.63 (2 dp) seconds for 11 peers. This is as expected, because the number of services should not influence the total average single query and response time.

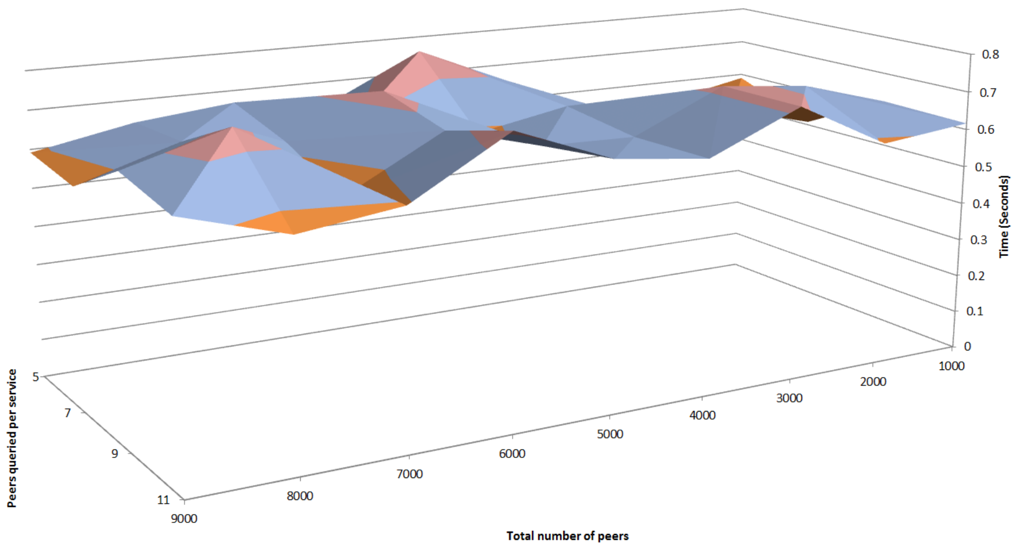

Figure 6.12: Single response time for $s = 3$.

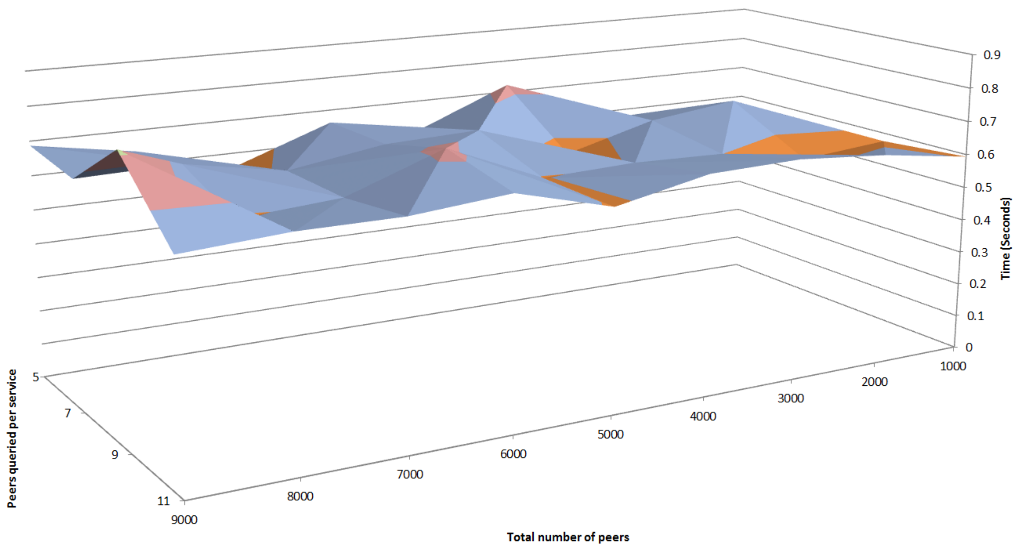

Figure 6.13: Single response time for $s = 5$.

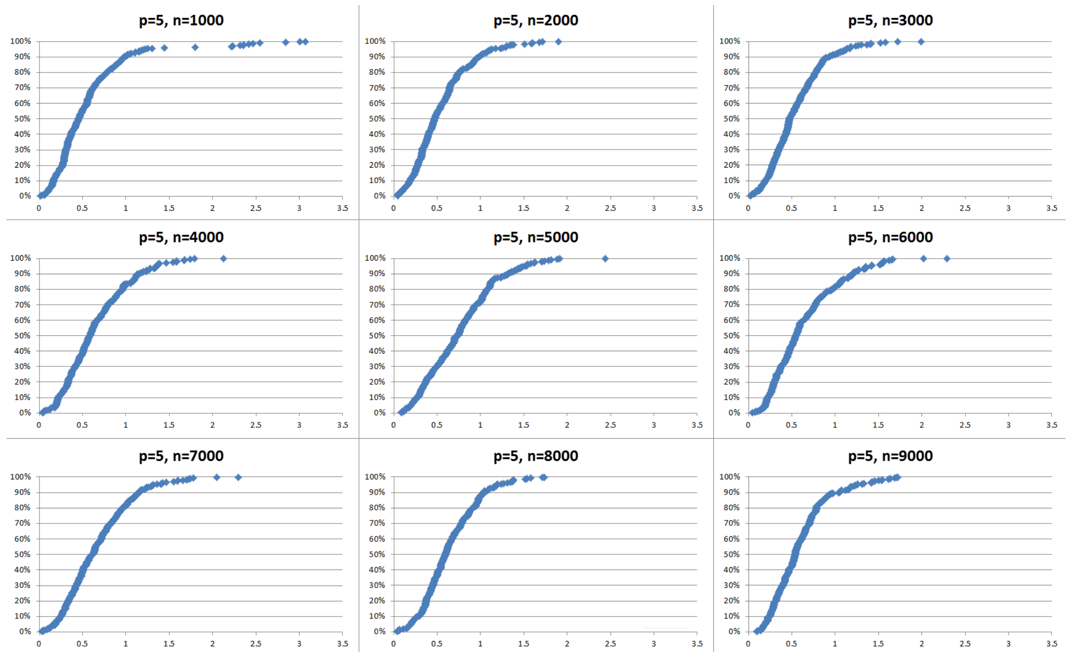

Figure 6.14: Cumulative frequency graphs for $s = 3, p = 5$.

Another way to examine this information is to consider it using a cumulative frequency graph. Figure 6.14 shows the nine cumulative frequency graphs (one for each n, from 1000 through to 9000) that make up the set for $s = 3, p = 5$. On the whole, the graphs within the set exhibit similar properties and have been combined together in all later cumulative frequency graphs.

Figure 6.15 shows the cumulative frequency graphs for $s = 3, p = 5, 7, 9, 11$. As can be seen from these graphs, they are essentially the same. However, there is a small trend towards later arriving responses as the value of p increases. A slight shift in the time taken to reach the 90% cumulative percentage is also apparent when going from $p = 5$ to $p = 11$. However, this cannot be seen on the graphs presented here.

6.7.2 Response Inter-arrival Times

The response inter-arrival time experiments record the time between subsequent responses arriving at the composer node. As all queries for profiles in the network are generated in parallel, the inter-arrival times should decrease as more queries are generated. This is because increasing the number of queries increases the number of responses received.

Figure 6.17 and Figure 6.18 show the inter-arrival times for query responses arriving at the composer node. As expected, the inter-arrival times decrease as more peers are queried. In other words, as q increases, either through increasing p or s, the inter-arrival time decreases. As 5 services require a larger number of queries than 3 services, it exhibits

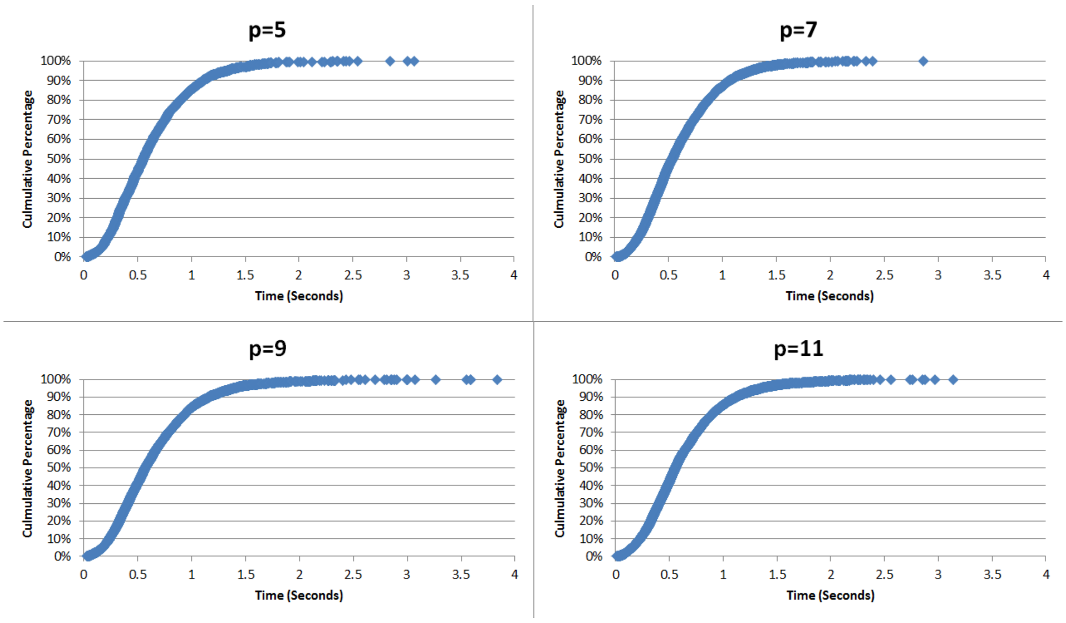

Figure 6.15: Combined cumulative frequency graphs for $s = 3, p = 5, 7, 9, 11$.

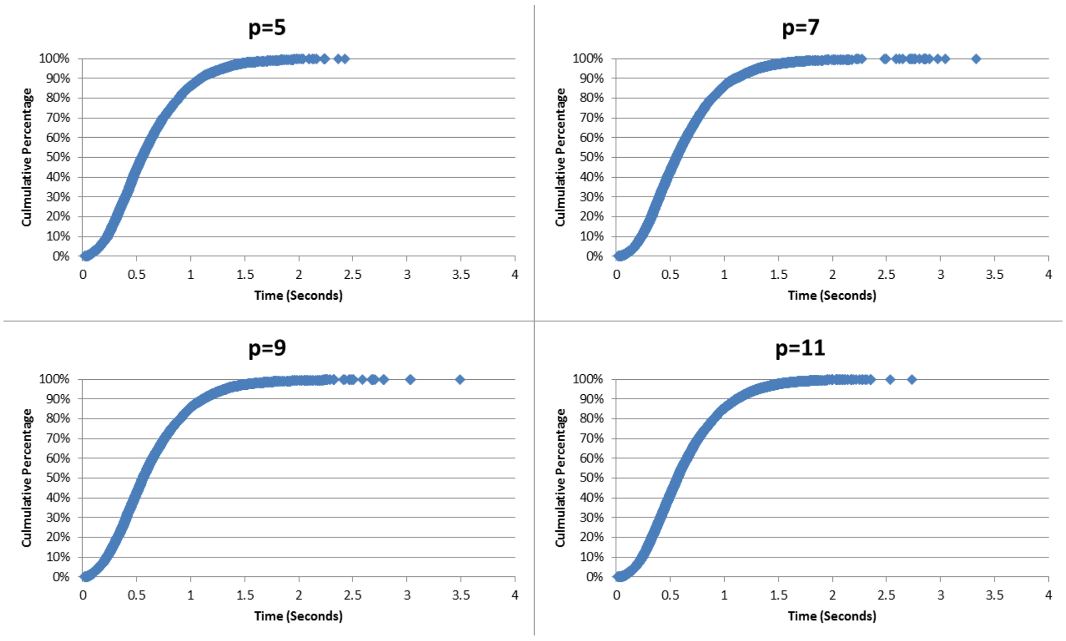

Figure 6.16: Combined cumulative frequency graphs for $s = 5, p = 5, 7, 9, 11$.

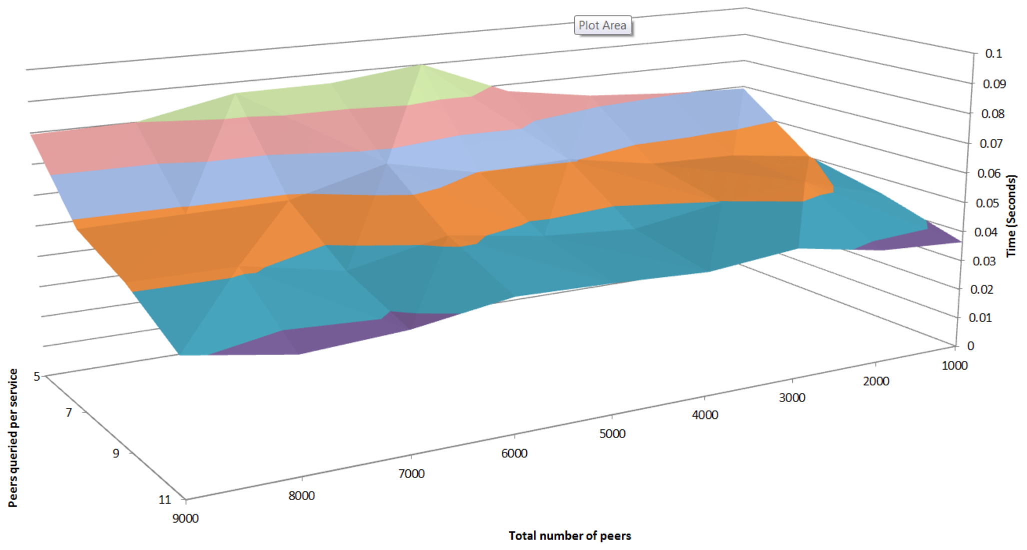

Figure 6.17: Response inter-arrival time for $s = 3$.

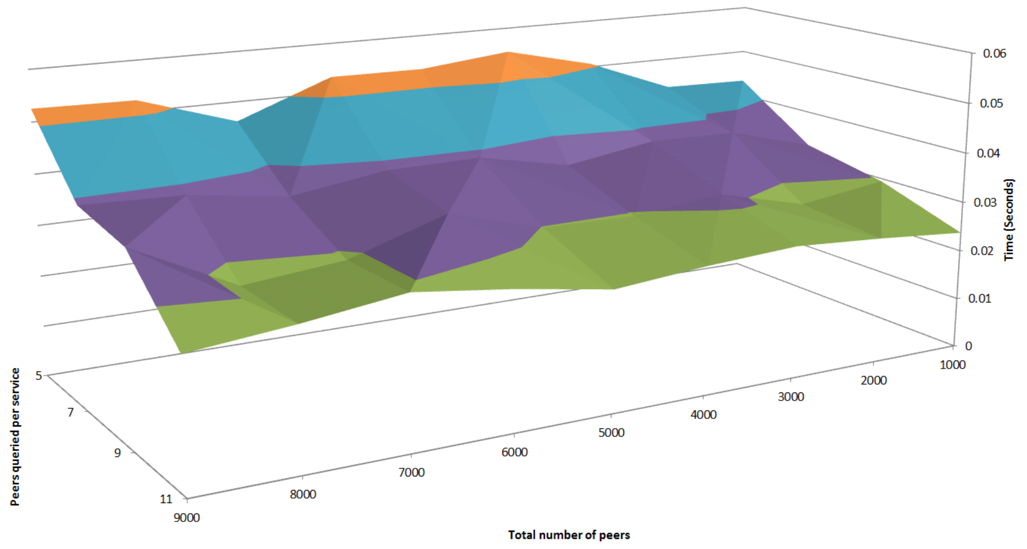

Figure 6.18: Response inter-arrival time for $s = 5$.

smaller inter-arrival times than the 3 services graph.

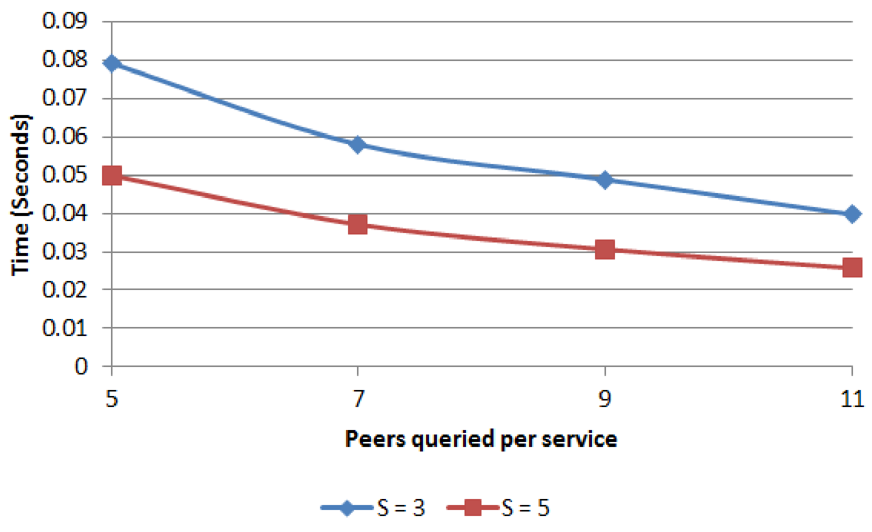

Figure 6.19: Response inter-arrival time for $s = 3$ and $s = 5$.

Figure 6.19 compares the inter-arrival times for $s = 3$ and $s = 5$. This graph verifies the earlier stated finding that increasing q decreases the inter-arrival time. This inter-arrival time graph is useful because it sets an approximate upper bound on the time between subsequent responses and hence determines a minimum performance requirement on any software that implements this approach. Inter-arrival times could also be used to determine the likely time to receive a given number of responses.

6.7.3 Total Time to Reach Threshold

The total time experiments record the total time taken to reach the $^1/_2$ and $^2/_3$ thresholds. Since there are no malicious peers, nor any churn, this is simply the time taken to return the first t responses out of the total q responses.

Figure 6.20 shows the total time taken to reach the threshold number of profiles for 3 services, with a threshold of $^1/_2$. In terms of the number of peers queried p, the total time increases from an average of 0.597 seconds (3 dp) for 5 peers, to an average of 0.602 seconds (3 dp) for 11 peers. When considering the total number of peers n, the total time ranges from an average of 0.56 seconds (2 dp) for 1000 peers through to 0.58 seconds (2 dp) for 9000 peers. On average, it can be said that the total time required to fetch all of the profiles increases as the number of peers queried p increases, and as the total number of peers n in the network increases.

Figure 6.21 shows the total time taken to reach the threshold number of profiles for 5 services, with a threshold of $^2/_3$. This graph exhibits the same behaviour as the graph for 3 services, but with the exception that the overall time has increased slightly as the

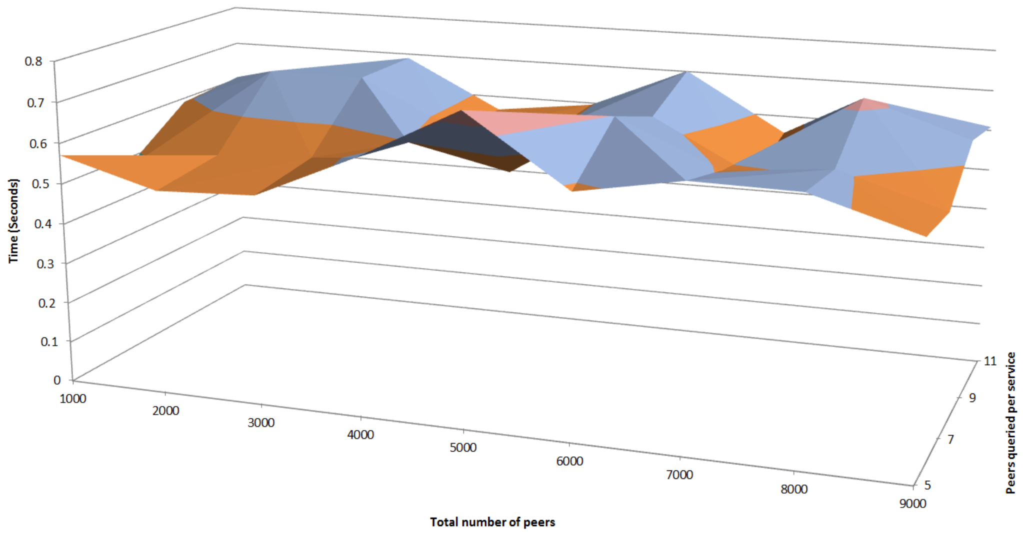

Figure 6.20: Total time for $s = 3$, $^1/_2$ threshold.

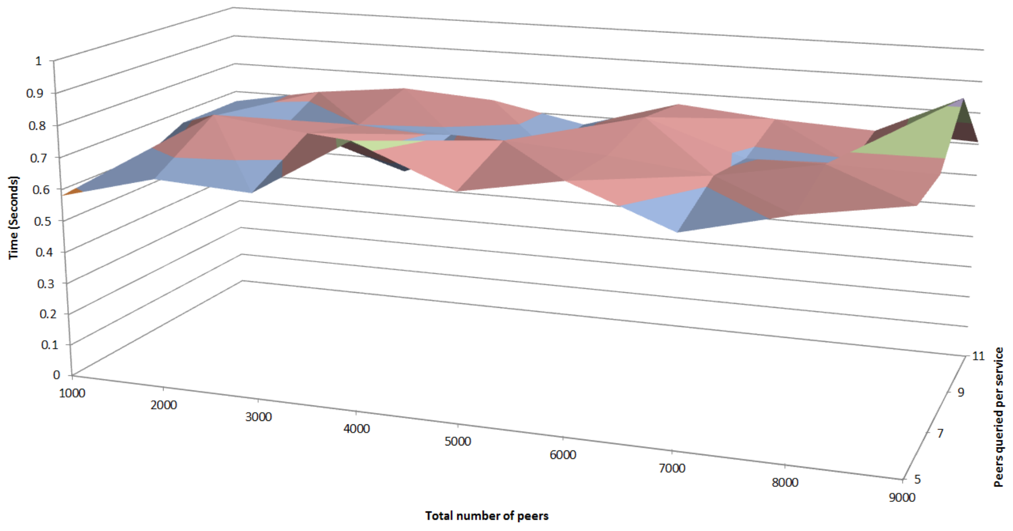

Figure 6.21: Total time for $s = 5$, $^2/_3$ threshold.

number of services s has increased from 3 to 5. In terms of the number of peers queried p, the total time taken increases from an average of 0.699 seconds (3 dp) for 5 peers, to an average of 0.719 seconds (3 dp) for 11 peers. Similarly, when considering the total number of peers n, the total time ranges from an average of 0.62 seconds (2 dp) for 1000 peers through to 0.79 seconds (2 dp) for 9000 peers.

Using this information and creating an average from the 3 and 5 service total times, Equation 6.3 describes a rule of thumb that can be used to estimate the total time, given the fraction f describing the threshold (for example, $f = 0.5$ when the threshold is $1/2$), and the total number of peers n.

$$time \approx 0.000022fn + 0.60 \tag{6.3}$$

In 2001, Ripeanu et al. [265] explored the then dominant GNUtella network. They discovered that at certain times, the GNUtella network contained almost 50,000 nodes. By 2004, file sharing via DHTs had become popular, and Kutzner et al. [186] found that the Overnet network had a maximum of approximately 265,000 concurrent nodes online. More recent estimates place the size of the mainline DHT that supports the BitTorrent network at about 15-27 million nodes online at any given time [322].

Figure 6.22 extrapolates the total time required to reach the threshold number of profiles for network sizes up to one million nodes using Equation 6.3. This size has been taken as a maximum as further extrapolation is likely to be increasingly inaccurate.

Section 6.2 discussed Nielsen's three web usability time limits. If Nielsen's 1 second limit were taken, then the GRAFT framework would be limited to approximately 60,000 nodes when using $1/2$ as the threshold. While this forms a useful baseline, it is not in itself helpful, as this limit would be impractical in most modern distributed networks. Therefore, the goal for an interactive session should be to return results to the user within the 10 second limit set by Nielsen. Rather than aiming for this limit, 5 seconds (being the halfway point between 1 and 10 seconds) will be used as the limit for any results returned to an interactive user. This limit is shown on the graph as a horizontal red line with two intersection points.

What this graph clearly shows is that for interactive sessions, networks over approximately 400,000 nodes (with a $1/2$ threshold) and 300,000 nodes (with a $2/3$ threshold) become impractical as the total time taken exceeds 5 seconds. However, it should be noted that users will tolerate much longer delays if they are kept informed. In particular, Nah [223] shows that users will, on average, tolerate delays of up to 38 seconds if they are provided feedback while waiting (2004). Therefore, increasing the time limit to 15 seconds and providing the users with feedback would allow GRAFT to exceed a million nodes. Further, the time limits discussed do not remove the overall usefulness for non-interactive systems where waiting time is not important, or where results can be cached

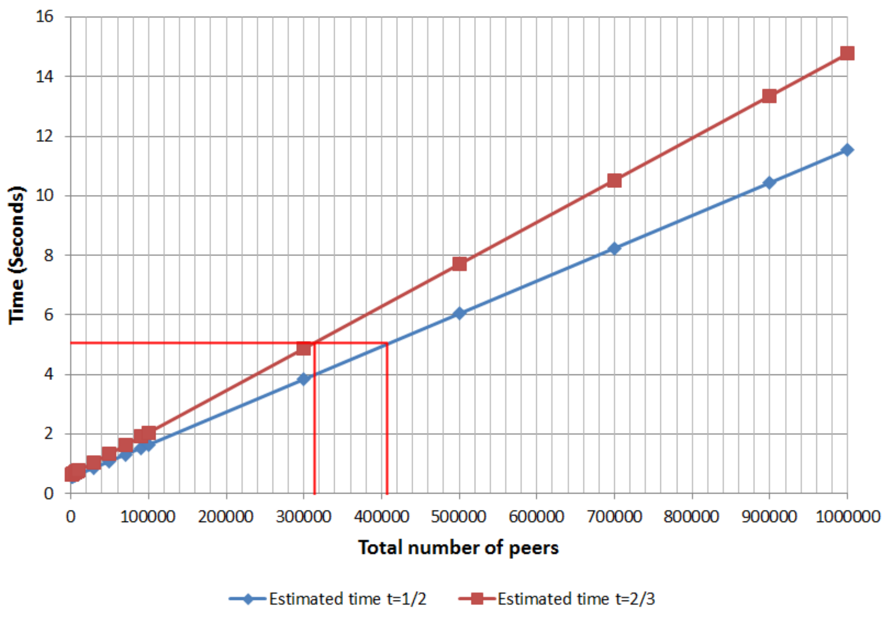

Figure 6.22: Total time taken versus total number of nodes. The total time required to reach the threshold number of profiles for network sizes up to one million nodes using Equation 6.3 is extrapolated.

ahead of time-sensitive operations.

While 400,000 or even a million nodes seem paltry when compared with those of the mainline DHT, there are a number of approaches that can be used to scale GRAFT further. These will be discussed in Section 6.11.

The work up to this point has assumed that the 1/2 threshold will continue to be effective, even when under churn and with malicious nodes. These aspects will be considered in the following sections.

6.8 Simulation: Base Case with Malicious Peers

The same experiments were run again, but with increasing rates of malicious peers. Malicious peers in these experiments return "corrupted" profiles, and so their responses do not match the responses from other peers. Threshold voting should eliminate these corrupted profiles, so it is useful to measure how well this mechanism works with malicious peers.

6.8.1 Design

In this experiment thresholds were ignored in the collection of the data, allowing for a complete view of the cross-over points between threshold voting succeeding and failing.

A goal of successful threshold voting with up to 30% malicious peers was set. In existing literature, values for malicious peers range between 0% and 70%, with typical default values being 10%, 20% and 30% [335, 364, 358, 61, 194, 290, 309]. Please note that these experiments do not consider collusive behaviour. This is left as future work.

6.8.2 Correct Responses

For these experiments, rate r controls the probability that any given peer is malicious. Values used for r in these experiments ranged from 0 through to 0.5. The maximum rate of 0.5 was used because rates higher than this always result in failure for the $1/2$ threshold (as it is impossible to get sufficient profile copies that are correct). The measure c counts the number of correct profiles that were received. For threshold voting to be successful, the number of correct profiles must be greater than or equal to the threshold, $c \geq t$.

Nodes	Malicious rate					
	0	0.1	0.2	0.3	0.4	0.5
1000	15	14	12	11	9	8
2000	15	14	13	11	10	8
3000	15	14	12	11	9	8
4000	15	14	13	12	10	9
5000	15	14	12	11	9	8
6000	15	14	12	10	9	7
7000	15	14	12	10	8	7
8000	15	14	12	10	9	7
9000	15	14	13	11	9	7

Table 6.1: Correct responses with malicious peers, $s = 3, p = 5$. The cell colours show the values that did not reach the $1/2$ threshold (8) in orange, while those not reaching the $2/3$ threshold (10) are shown in yellow.

Table 6.1 shows the number of correct responses for both thresholds, for 3 services. The integer floor value is shown in each instance, as it is not possible to have partial profiles. The cell colours show the values that did not reach the $1/2$ threshold (8) in orange, while those not reaching the $2/3$ threshold (10) are shown in yellow. Figure 6.23 is a graphical representation of the same information, with variables n (total number of peers), r (rate of malicious peers) and c (correct responses) shown as axes on the graph.

Table 6.2 shows the number of correct responses for both thresholds, for 5 services. The cell colours show the values that did not reach the $1/2$ threshold (28) in orange, while those not reaching the $2/3$ threshold (37) are shown in yellow. Figure 6.24 is a graphical

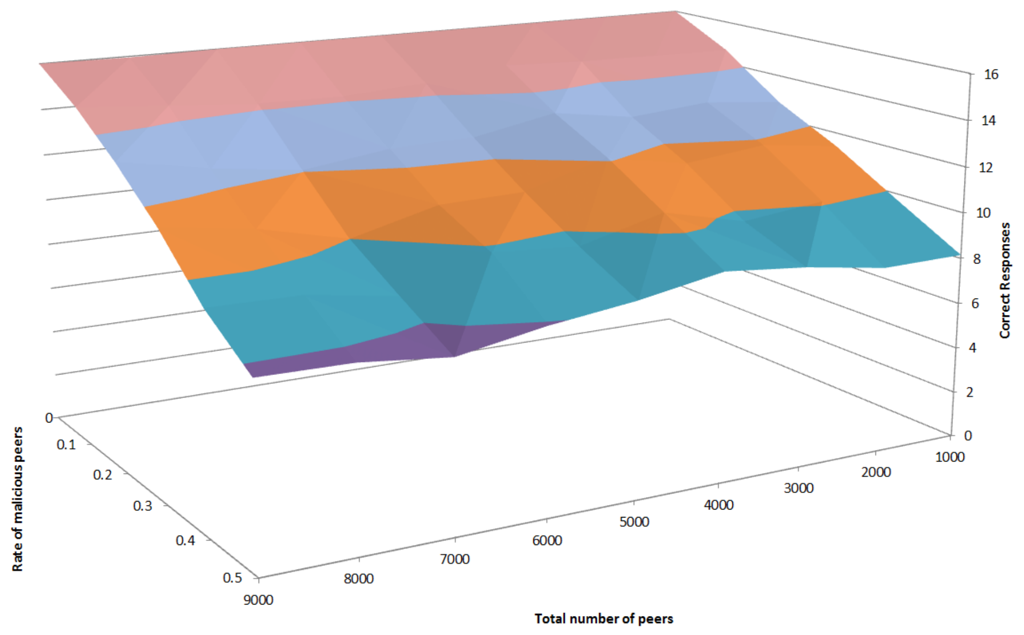

Figure 6.23: Correct responses with malicious peers, $s = 3, p = 5$. A graphical representation of Table 6.1.

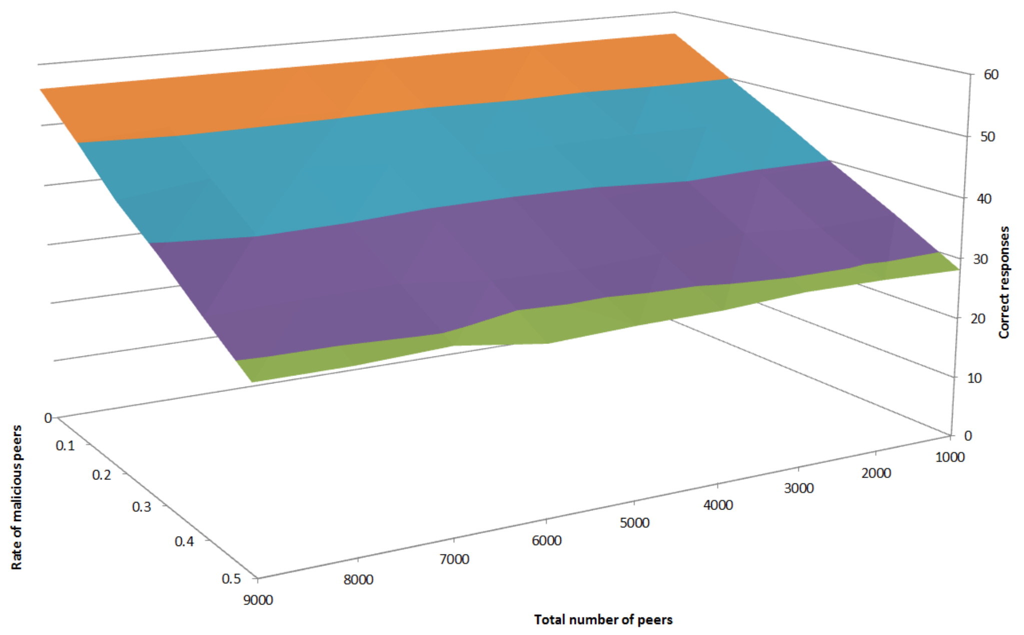

Figure 6.24: Correct responses with malicious peers, $s = 5, p = 11$. A graphical representation of Table 6.2.

Nodes	Malicious rate					
nodes	0	0.1	0.2	0.3	0.4	0.5
1000	55	50	45	39	34	28
2000	55	50	44	39	33	28
3000	55	50	44	40	34	28
4000	55	50	44	38	33	27
5000	55	50	44	38	32	27
6000	55	50	44	38	32	27
7000	55	50	45	39	34	28
8000	55	50	45	40	33	28
9000	55	50	43	39	33	28

Table 6.2: Correct responses with malicious peers, $s = 5, p = 11$. The cell colours show the values that did not reach the $1/2$ threshold (28) in orange, while those not reaching the $2/3$ threshold (37) are shown in yellow.

representation of the same information.

In the perfect situation described by the base case, the tables show that threshold voting works with a malicious rate of up to 40% with the $1/2$ threshold, while the $2/3$ threshold works well up to a malicious rate of 30%. Both of the graphs show a similar trend, with a decreasing number of correct responses c as the rate r of malicious peers increases.

6.8.3 Trend

s	p	q	t	Equation
03	05	15	08	c = -15.84r + 16
03	07	21	11	c = -21.72r + 22
03	09	27	14	c = -27.78r + 28
03	11	33	17	c = -33.66r + 34

Table 6.3: Trend analysis for correct responses. This table shows part of the analysis that was conducted (in this instance the values relate to $s = 3$). All values are rounded to 2 decimal places.

When examining all of the correct response results for variations in s and p, a trend became apparent. The trend is linear, and in general, the coefficient of the equation is approaching the total queries q. Table 6.3 shows part of the analysis that was conducted (in this instance the values relate to $s = 3$). All values are rounded to 2 decimal places.

Equation 6.4 describes a rule of thumb that can be used to estimate the number of correct responses c when given the total queries q and the malicious rate r. Building on this, Equation 6.5 describes a rule of thumb that can be used to determine the boolean result b: "given c, will the threshold t will be met or exceeded?".

$$c \approx \lfloor q - rq \rfloor \tag{6.4}$$

$$b = \begin{cases} true & \text{if } t \geq c \\ false & \text{otherwise} \end{cases} \tag{6.5}$$

6.9 Simulation: Churn

Churn is generated using a Pareto function, with the mean session time set to 100 seconds. A Pareto function was selected because this mimics real-world measurements taken on the BitTorrent network [146], while 100 seconds was selected as Rhea et al. [264] observed mean session times ranging from one minute up to one hour. Measuring churn using smaller session time means that the experiments will capture the worst-case behaviour. In particular, Herrera et al. [140] discuss how increasing membership fluctuations in a DHT lead to decreasing performance. All other parameters are left unchanged from previous experiments. As with previous experiments, each experiment ran for 9000 seconds simulated time, and was repeated 40 times.

6.9.1 Single Response Time

In the same way as the single response time experiment in the base case, this experiment records the average time taken to send a query for a profile and receive a response.

Figure 6.25 shows the cumulative frequency graphs for $s = 3, p = 5, 7, 9, 11$, while Figure 6.26 shows the cumulative frequency graphs for $s = 5, p = 5, 7, 9, 11$. Compared to the earlier combined cumulative frequency graphs in Figure 6.15 and Figure 6.16, these graphs cover a time-period that is more than twice as long (9 seconds instead of the previous 4 seconds). However, it should be noted that once again, 90% of the responses have arrived by approximately 1 second. The key difference is the number of responses that arrive much later than in the earlier experiments.

6.9.2 Response Inter-arrival Times

As with the base case, this experiment records the time between subsequent responses arriving at the composer node. Since the composer issues queries in parallel, the inter-

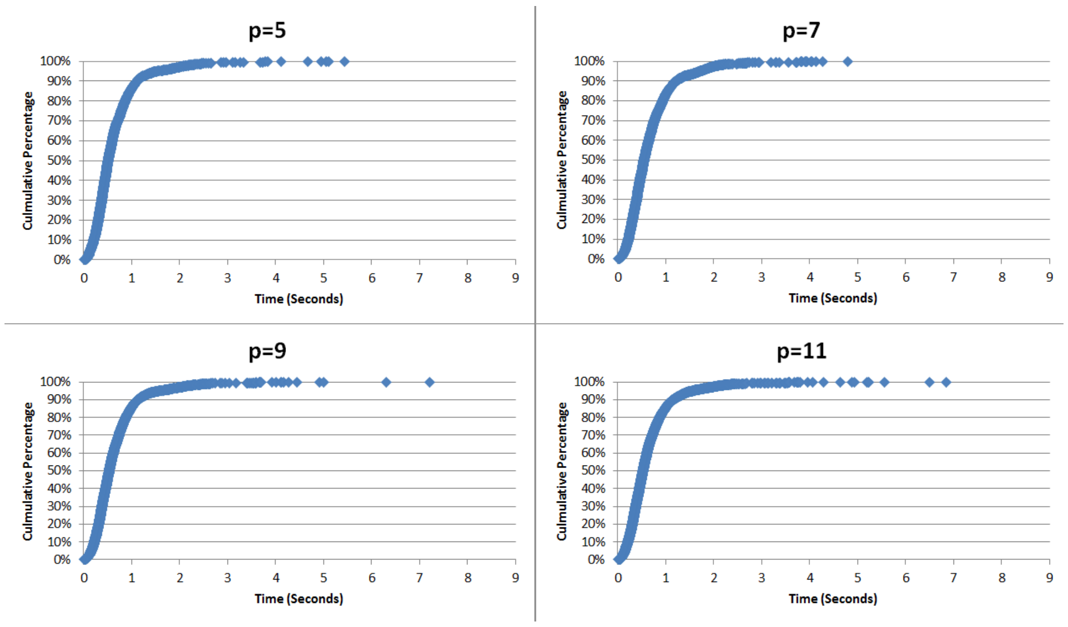

Figure 6.25: Combined cumulative frequency graphs for $s = 3, p = 5, 7, 9, 11$ and churn.

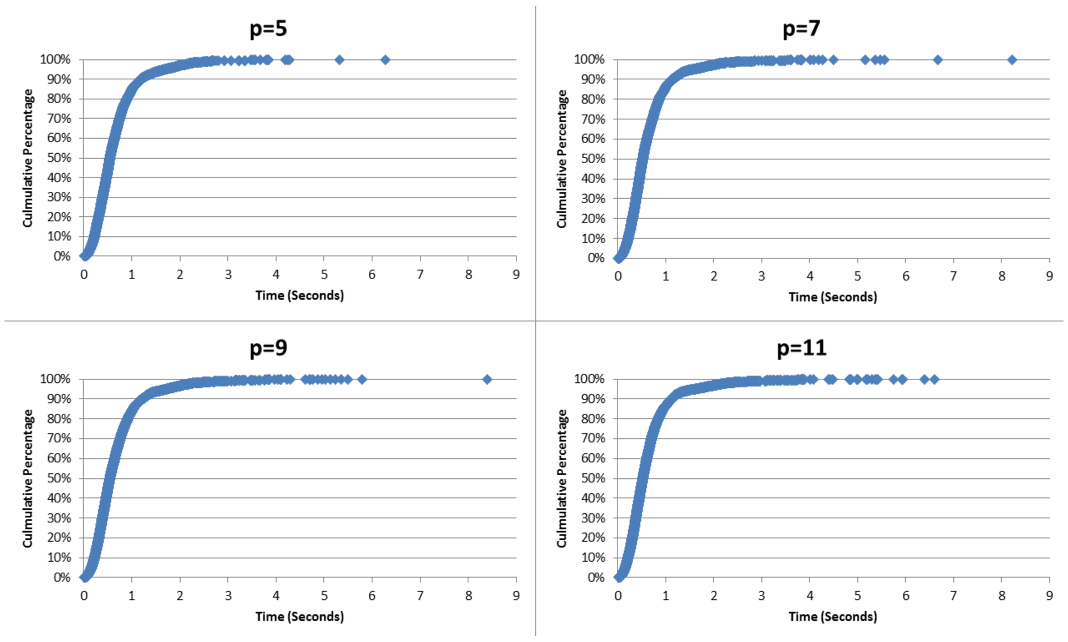

Figure 6.26: Combined cumulative frequency graphs for $s = 5, p = 5, 7, 9, 11$ and churn.

arrival times should decrease as more queries are generated.

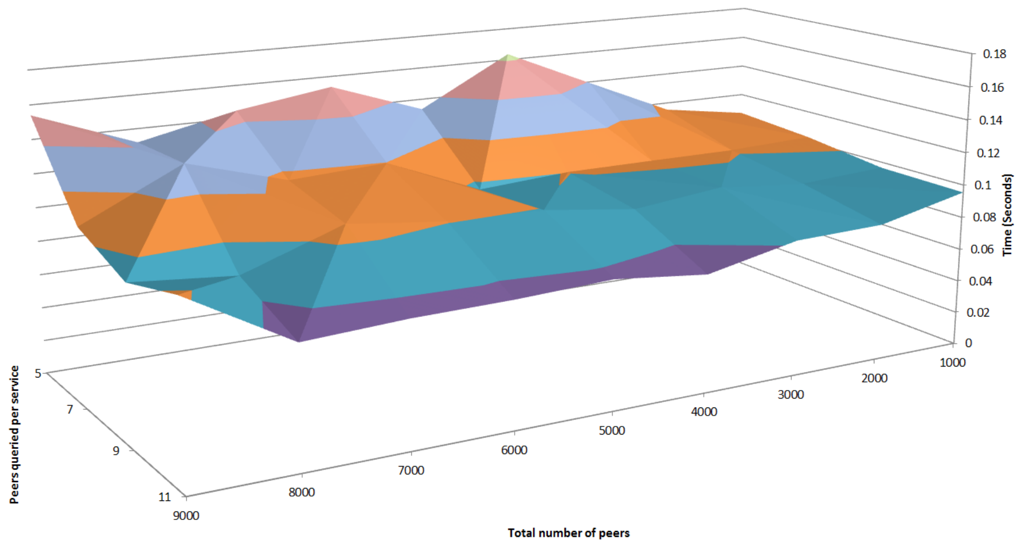

Figure 6.27: Response inter-arrival time for $s = 3$ with churn.

Figure 6.29 compares the inter-arrival times for $s = 3$ and $s = 5$ when under churn. This graph verifies the earlier stated finding that increasing q decreases the inter-arrival time even when under churn. It is worth noting that the inter-arrival times in this graph are approximately double the values present in Figure 6.19.

6.9.3 Total Time to Reach Threshold

Similar to the total time experiment in the base case, this experiment records the total time taken to reach the threshold. As churn may affect the number of responses, a 5 second time limit exists to ensure that the composer node is not left waiting indefinitely. In those instances where a threshold number of responses was not received with the time limit, then the total time to reach the threshold has been recorded as 5 seconds. This is because after hitting the time limit, the composer does not have sufficient responses to reach a valid conclusion.

Figure 6.30 shows the total time taken to reach the threshold number of profiles for 3 services, with a threshold of $^1/_2$. In terms of the number of peers queried p, the total time decreases from an average of 0.763 seconds (3 dp) for 5 peers, to an average of 0.672 seconds (3 dp) for 11 peers. When considering the total number of peers n, the total time ranges from an average of 0.60 seconds (2 dp) for 1000 peers through to 0.73 seconds (2 dp) for 9000 peers. On average, it can be said that the total time required to fetch all of the profiles increases as the number of peers queried p increases, and as the total number of peers n in the network increases.

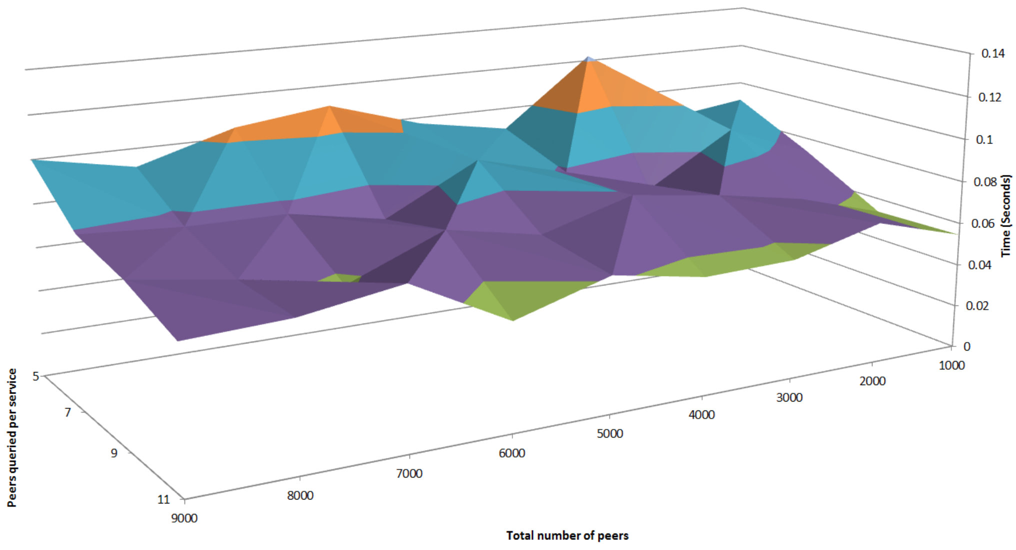

Figure 6.28: Response inter-arrival time for $s = 5$ with churn.

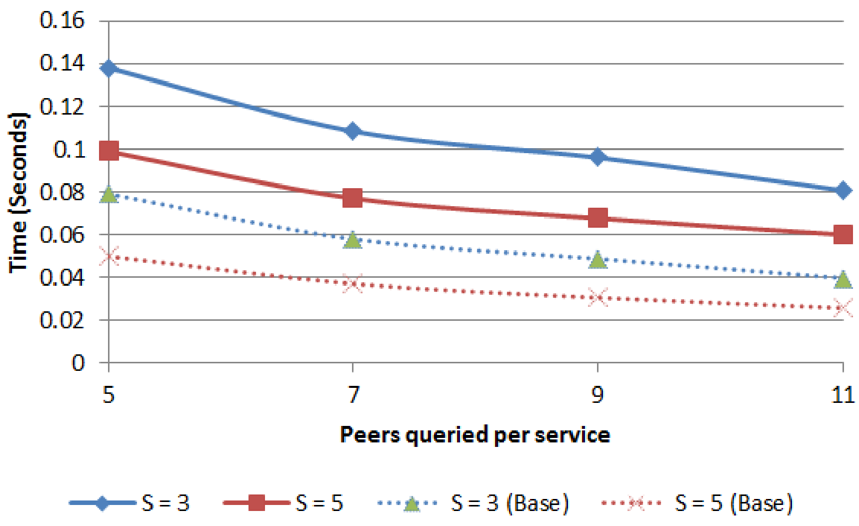

Figure 6.29: Response inter-arrival time for $s = 3$ and $s = 5$ with churn, base case values are shown here with dashed lines for comparison.

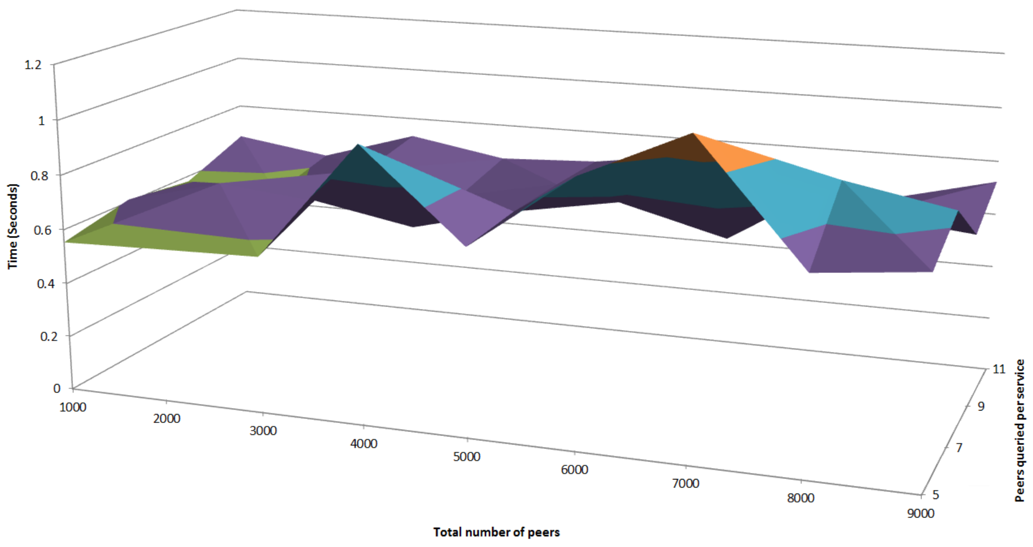

Figure 6.30: Total time for $s = 3$ with churn, $1/2$ threshold.

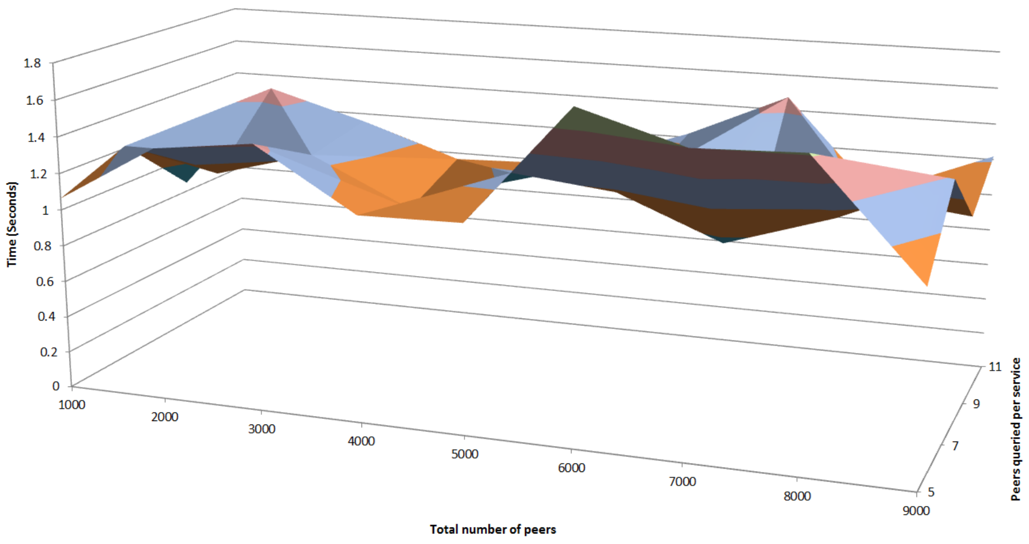

Figure 6.31: Total time for $s = 5$ with churn, $2/3$ threshold.

Figure 6.31 shows the total time taken to reach the threshold number of profiles for 5 services, with a threshold of $2/3$. This graph exhibits the same behaviour as the graph for 3 services, but with the exception that the overall time has increased slightly as the number of services s has increased from 3 to 5. In terms of the number of peers queried p, the total time taken decreases from an average of 1.349 seconds (3 dp) for 5 peers, to an average of 1.099 seconds (3 dp) for 11 peers. Similarly, when considering the total number of peers n, the total time ranges from an average of 1.06 seconds (2 dp) for 1000 peers through to 1.17 seconds (2 dp) for 9000 peers.

6.10 Simulation: Churn with Malicious Peers

The same churn experiments were run again, but with increasing rates of malicious peers.

6.10.1 Correct Responses

As in Section 6.8, the rate r controls the probability that any given peer is malicious, while t is the threshold. c counts the number of correct profiles received. A successful threshold vote occurs only when $c \geq t$.

Nodes	Malicious rate					
	0	0.1	0.2	0.3	0.4	0.5
1000	12	11	10	9	7	6
2000	13	12	11	10	8	6
3000	13	12	10	9	8	7
4000	12	11	10	9	8	7
5000	12	11	10	8	7	6
6000	13	11	10	8	7	6
7000	12	11	9	7	6	5
8000	12	11	9	8	7	5
9000	12	11	10	9	7	5

Table 6.4: Correct responses with churn and malicious peers, $s = 3, p = 5$. The cell colours show the values that did not reach the $1/2$ threshold (8) in orange, while those not reaching the $2/3$ threshold (10) are shown in yellow.

Table 6.4 shows the number of correct responses for both thresholds, for 3 services. All values have been rounded down, as it is not possible to have partial profiles. The cell colours show the values that did not reach the $1/2$ threshold (8) in orange, while

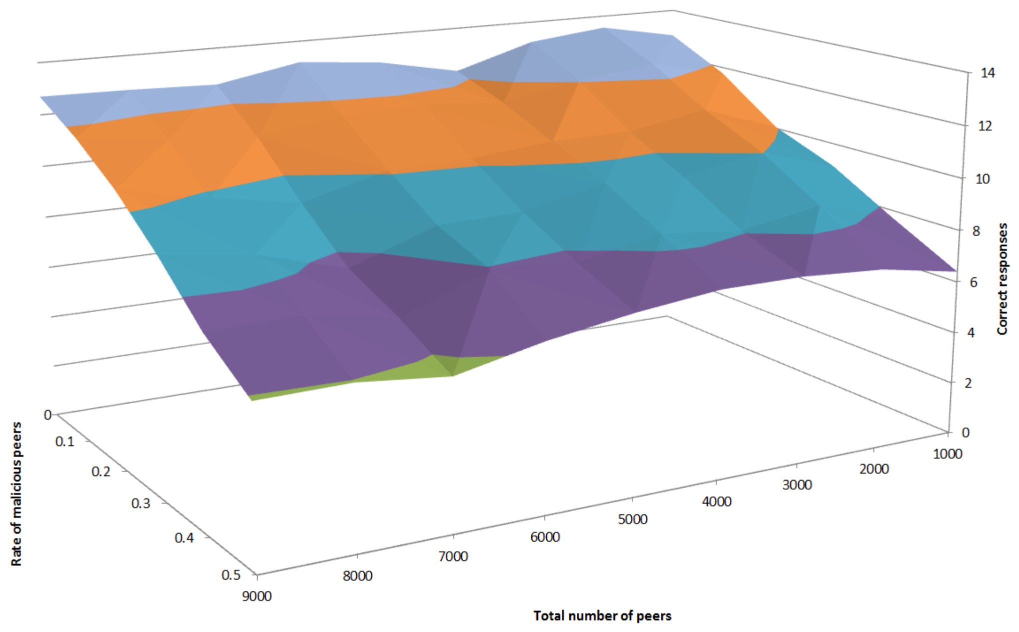

Figure 6.32: Correct responses with churn and malicious peers, $s = 3, p = 5$. A graphical representation of Table 6.4.

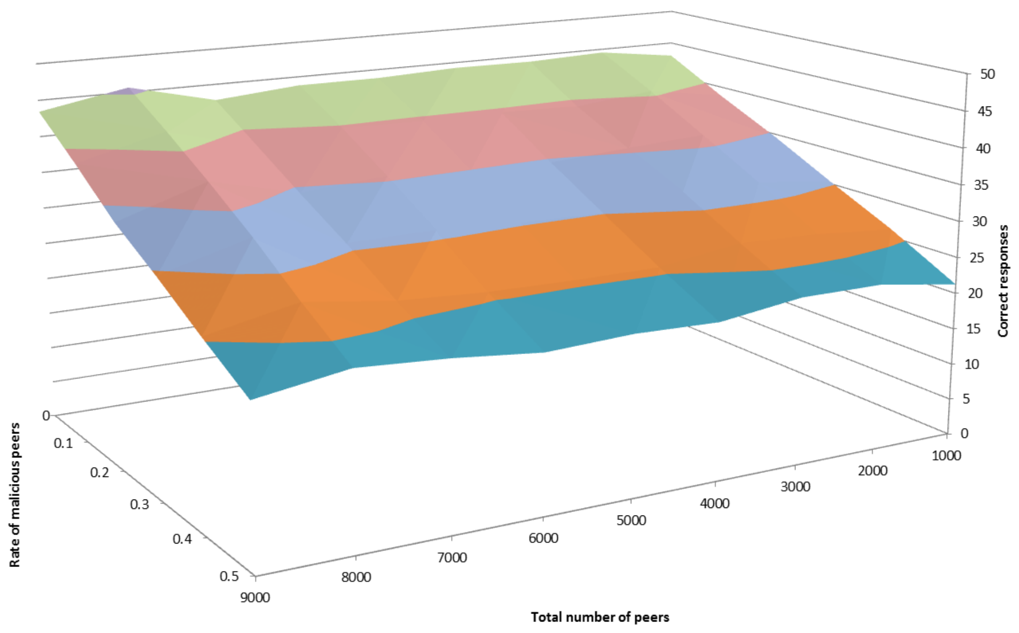

Figure 6.33: Correct responses with churn and malicious peers, $s = 5, p = 11$. A graphical representation of Table 6.5.

Nodes	Malicious rate					
	0	0.1	0.2	0.3	0.4	0.5
1000	43	38	34	30	26	21
2000	44	40	36	31	26	22
3000	44	39	35	31	27	22
4000	44	39	34	30	25	21
5000	43	39	35	30	25	21
6000	43	39	35	30	25	21
7000	42	38	34	30	26	22
8000	45	41	37	32	27	23
9000	43	38	33	29	25	21

Table 6.5: Correct responses with churn and malicious peers, $s = 5, p = 11$. The cell colours show the values that did not reach the $1/2$ threshold (28) in orange, while those not reaching the $2/3$ threshold (37) are shown in yellow.

those not reaching the $2/3$ threshold (10) are shown in yellow. Figure 6.32 is a graphical representation of the same information.

Table 6.5 shows the number of correct responses for both thresholds, for 5 services. The cell colours show the values that did not reach the $1/2$ threshold (28) in orange, while those not reaching the $2/3$ threshold (37) are shown in yellow. Figure 6.33 is a graphical representation of the same information.

As in Section 6.8, the tables show that threshold voting is working. However, the addition of churn to these experiments has reduced the overall effectiveness of the threshold voting. The $1/2$ threshold is now effective up to a malicious rate of 30%, while the $2/3$ threshold is now only effective up to a malicious rate of 10%. The latter suggests that the $2/3$ threshold is not practical in real-world situations, leaving only the $1/2$ threshold. Alternatively, both thresholds could be used, with the $1/2$ acting as an absolute cut-off, leaving the other threshold acting only as an untrusted "benchmark".

6.10.2 Trend

Similarly to Section 6.8, a trend was apparent when examining all of the correct response results for variations in s and p. An analysis showed that the trend was linear, and that the coefficient of the equation was a multiple of q. Table 6.6 shows part of the analysis that was conducted (in this instance the values relate to $s = 3$). All values are rounded to 2 decimal places.

In the table, the coefficient expresses a value of approximately $0.85q$ (2 dp). To this

s	p	q	t	Equation
03	05	15	08	c = -12.80r + 12.85
03	07	21	11	c = -17.06r + 17.28
03	09	27	14	c = -22.12r + 22.43
03	11	33	17	c = -26.66r + 27.02

Table 6.6: Trend analysis for correct responses when under churn. This table shows part of the analysis that was conducted (in this instance the values relate to $s = 3$). All values are rounded to 2 decimal places.

end, Equation 6.6 describes a rule of thumb that can be used to estimate the number of correct responses c when under churn, and given the total queries q and the malicious rate r.

$$c \approx \lfloor 0.85q - rq \rfloor \tag{6.6}$$

6.11 Summary

The experiments conducted in this chapter show that the architecture is robust and performs as designed. The introduction to this chapter considered the evaluation as containing three distinct aspects: the performance of recommendation sources; the ability to utilise the architecture to provide services; and finally, the ability to scale the architecture and efficiently provide and verify profiles.

The operation and performance of the source nodes is critical to the architecture. The experiments show that the source nodes are feasible, but only for non-interactive usage. In particular, the recommendation source based on co-authorship relationships takes too long to fetch and compute those relationships. Pre-computing and storing the co-author relationship information is feasible up to the 3rd degree when using profile compression. This clearly favours a non-interactive approach to generating and storing the information in profiles. Similarly, the H-index source experienced poor worst-case performance. Based on these results, a non-interactive approach also makes sense for this source. The usage of a non-interactive approach matches the original design intent of the architecture. Of particular note is that the non-realtime nature of these updates does not affect the ability for their respective sources to generate regular updates to profiles.

The ability to utilise the architecture to provide services without significant overheads being introduced was also important. The experiments used a ranking service example to demonstrate that the overhead of utilising such a service was minimal.

Finally, the experiments show that the architecture handles both churn and malicious peers, and that it can scale with some limitations. The $1/2$ threshold has been shown to work with up to 30% malicious peers while under churn. The simulations also show that the $2/3$ threshold is impractical in real-world situations, as it can only handle a maximum 10% malicious peer rate when churn is present. Extrapolation from the simulation results show that when limited to a maximum of 5 seconds, and noting that the experiments are targetted at worst-case performance, the network can scale up to 400,000 nodes using the $1/2$ threshold. Increasing the time limit to 15 seconds will allow the network to grow to a million nodes.

With modern networks scaling to millions of nodes online concurrently, a maximum of one million nodes is potentially limiting. However, scaling the network to accommodate millions of clients can be accomplished without changing the architecture of the framework.

Firstly, removing all of Nielsen's user experience time-limits would allow the network to grow much larger, as has already been demonstrated when increasing the time-limit from 5 to 15 seconds. As discussed in Nah [223], user experience design will go a long way towards ameliorating user frustration in waiting for results.

Secondly, the widespread use of agent nodes that act on behalf of non-GRAFT clients would allow the network to grow significantly. Exactly like the ranking service described in Section 4.2.1, these agent nodes would provide recommendation services to the community. Given that any one agent could service a large number of clients, this would allow the network to scale significantly, also without any design changes. Assuming a 1:100 ratio of agents to clients, the "core" network could reasonably scale to handle over a 100 million clients.

Thirdly, one of the use cases for GRAFT is in the bootstrapping of new users. For example, in modern forums, the user registration process often results in an email confirmation being sent to validate the email address before granting full access. This registration can sometimes take anywhere from a few minutes to a few hours, as users often need to wait for the verification email to arrive. Using this time to fetch and verify a user's profile in parallel would obviate any waiting time for that user.

Finally, extensive use of profile caching in services (such as agents), as demonstrated again by the ranking service, would also remove some of the the waiting time. In particular, many services are not time critical, or the results can be pre-fetched and cached for later use.

In summary, the source nodes are feasible as originally intended and designed. Adopting the GRAFT framework within a service is feasible because using GRAFT does not impose significant performance overheads. Finally, although the "core" network is limited, this does not impact on the ability for the architecture to service millions of clients.

Chapter 7

Conclusions

This thesis has discussed the design, prototype implementation and evaluation of the GRAFT framework. GRAFT was designed to meet five clear goals. These goals are restated here from Section 1.3.

G.1 The use of recommendation information derived from non-reputation sources, and the subsequent usage of this information within a reputation system.

G.2 The integration of human and electronic entities, including the investigation of how these could be combined into one recommendation system.

G.3 The integration of identity and reputation, with a focus on how reputation information could be combined with identity information.

G.4 The use of contextual information within reputation systems. In particular, the ability to capture and make this information available for use in the evaluation of recommendations.

G.5 The ability to exchange recommendation information between systems.

The architecture for GRAFT achieves all of the stated requirements. This chapter reviews the goals, discusses contributions made by this thesis and presents possible future work.

7.1 Review

GRAFT is a generalised framework that supports the collection and distribution of recommendation information for both people and electronic service entities. The recommendation information is obtained from multiple sources, and fed continuously into GRAFT. This is then made available to consumers. Recommendation information in GRAFT is stored in structured documents called profiles, with every profile holding one or more recommendations, each from a unique source. The recommendations that make up a profile support multiple contexts, and contextual attributes.

A profile is located using the OpenID of an entity. All entities within GRAFT have an OpenID, which acts as both the entity's identity, and as a key to locate their recommendation profile. In the lightweight model, the location of the profile is discovered using the OpenID of the user, while in the integrated model, additional copies of the profile are located by using the OpenID as a key within a Kademlia DHT.

Each source of recommendation information is made up of either explicit, or implicit, reputation information. A number of different sources of recommendation are discussed in this thesis, including two sources derived from non-reputation information. In particular, a source that utilises Hirsch Index (H-index) information and a source that builds upon co-author relationship information. However, there are no explicit limits on the types of information that could be fed into GRAFT profiles.

The consumers utilise the information obtained from the sources to either make decisions (using policies) about individual entities, or to pass the information on to a third-party consumer, such as in the case of the ranking service. The exact policy implemented by each consumer may be unique and is only limited by the information made available by the sources. All of the sources and consumers are nodes in a single Kademlia DHT. Each node may be asked to store a number of profiles for GRAFT. Rather than store each profile in the network only once, they are each replicated multiple times for both reliability and resilience against malicious nodes.

7.1.1 Implementation

The GRAFT proof of concept prototype introduced in Chapter 5 includes functional implementations of all of the major components of the framework. To demonstrate the versatility of the GRAFT framework, three of the case studies presented in Chapter 4 were implemented. These included a forum, a wiki and a workflow implementation.

In the forum implementation, recommendation information from one forum was fed to another in order to augment a user's reputation or bootstrap it where the user was new. This case study demonstrated that GRAFT could be used to transfer recommendation information from one system to another, and that it could also be used to cache recommendation information for a single consumer/source system.

The wiki implementation used the H-index and co-author relationship sources to implement access control on wiki pages. This allowed the page creator to limit access to a page based on both the "degrees of co-authorship" relationship to the user and their academic standing as measured using H-index.

Finally, the workflow implementation added a ranking service and two additional components to the Anduril workflow engine. The two components allowed a workflow to request a list of web services (matching a given context) from the ranking service, and get back a sorted list of applicable web services. The workflow was then able to call the highest ranked web service.

7.1.2 Experimental Results and Evaluation

The experimental results and evaluation presented in Chapter 6 show that the GRAFT framework is robust and performs as designed. Three distinct aspects of the architecture were evaluated.

Firstly, the two novel recommendation sources were evaluated for their performance. The evaluation showed that both sources can take significant periods of time to compute a result and as such are suited only to non-interactive profile updates. However, this aligns exactly with the original design intent of the architecture.

Secondly, the ability to utilise the framework without introducing significant overheads was measured. The evaluation showed that a basic ranking service implementation added only minimal overhead.

Finally, the third aspect considered the effects of both churn and malicious peers upon the peer-to-peer network and the ability for the architecture to scale. The architecture was able to handle both churn and up to 30% malicious peers. A user experience time limit meant that the network could only scale to 400,000 nodes. However, removing that limit and adopting some optimisations would allow the architecture to scale successfully to handle over a 100 million clients.

7.2 Contributions

This thesis makes a number of contributions in the areas of reputation, reputation systems and access control. The following contributions are re-stated here from Section 1.4. This thesis:

1. Defines a terminology for reputation that describes reputation, trust and risk and then discusses the relationships between them. The term recommendation as it is used in the context of this thesis is also defined and discussed. This contribution can be found in Section 2.1.

2. Defines a generalised reputation model that can be used to describe reputation in both online and offline contexts. This model introduces standardised terminology (based on published research) that is used throughout this thesis. This contribution can be found in Section 2.2.

3. Defines an individual reputation context model that describes all of the contexts that an individual entity may possess. This model is useful in that it helps to describe the contextual nature of an individual and makes clear the need for multiple context support in reputation systems. This contribution can be found in Section 2.2.

4. Provides a survey and taxonomy for reputation systems. The survey, conducted on both academic and commercial systems, was used to drive the development of the taxonomy. The taxonomy builds on five commonly accepted dimensions, while also providing nine new dimensions for those aspects of reputation systems that had not been covered widely previously, or never considered before. The taxonomy is subsequently used to build a classification of existing reputation systems, generating a large number of research leads. These contributions can be found in Section 2.3.

5. Presents an architecture (Chapter 3) and prototype (Chapter 5) of a fully distributed recommendation system that supports both human and electronic service entities. The architecture and prototype encapsulate a paradigm shift in reputation systems and exhibit a number of unique ideas:

   (a) A novel and well-defined three layer architecture stack that underpins the design and implementation of GRAFT. The stack allows for a modular approach to building and integrating components into GRAFT, simplifying new design and development work. In particular, the separation of raw collection from integration into the peer-to-peer network allows for efficient nodes that only implement those aspects of the stack they require.

   (b) An exploration of identity and reputation integration. All recommendation information in GRAFT is tied to identity information. Knowing the identity of an entity is sufficient to be able to locate and consume their recommendation information. Previous systems have treated these two concepts as distinct, leading to classic two-step authenticate and authorise models. GRAFT instead considers all information about an entity when granting access, leading to better decisions.

   (c) The integration of both human and electronic service entities into one reputation system. These two types of entities are treated identically within GRAFT. Their integration into one reputation system is possible due to the fact that identity and recommendation information have been combined using OpenID.

   (d) The utilisation of a peer-to-peer network to store and replicate recommendation information. The peer-to-peer network distributes the load evenly across the peers that make up the network, whilst also ensuring robustness for profiles.

(e) The utilisation of both explicit and implicit reputation information. In particular, the use of information previously not regarded as being useful when combined with reputation such as demographic, social and derived information.

(f) The ability to retain the context in which reputation information was generated. Understanding the context of the information allows for it to be utilised in a meaningful way by a consumer.

(g) The utilisation of recommendation information in policy description and evaluation. The ability to combine recommendation information from multiple sources in the building of policies allows for flexibility that is otherwise not possible. For example, policies can build on the user's demographics and their professional standing, but can also utilise their social relationships.

6. Analysis of the performance of the GRAFT architecture. In particular, the performance is measured and significant factors affecting performance are identified using a series of experiments that utilise both the prototype and large-scale simulations. The simulations consider both the "perfect" state and increasing levels of churn and malicious peers. These contributions can be found in Chapter 6.

7.3 Future Work

The goals and scope stated in Chapter 1 purposely limited the work that would be addressed in this thesis. However, a number of areas for future work were identified during the writing of this thesis and are documented here.

7.3.1 Architecture

Agent nodes

Given the limitations of the architecture, further research should consider the widespread use of "agents". Agent nodes would be fully integrated into the network, but act on behalf of requestors. Much like the ranking service described in Section 4.2.1, these agents would provide recommendation services to the community. Given that an agent could service a large number of clients, this would allow the network to scale significantly. Research is required to determine the scability of agents, and how profile caching affects both their performance and the accuracy of their results.

Alternative Frameworks

One of the key requirements that drove the GRAFT design was that it was open and based on existing standards. OpenID is an open and standard approach to distributed identity on the Internet. As a key component of the framework, OpenID was critical to ensuring that identity and reputation could be tied together. An aspect that was not considered or explored was the use of identity frameworks other than OpenID. Commercial frameworks were ruled out because of their closed nature. However, there may be other frameworks that could be used which may confer additional advantages. Further work is required to establish if other identity frameworks could be used, and if these would behave in a similar fashion to OpenID, or if compromises would be required.

7.3.2 Profiles

Validation

Within this thesis, profile validation was performed using threshold voting. Further work is required to determine if other approaches would work when validating profiles, and if these might be more efficient. For example, it might be possible to store only checksums, and use these to validate a profile, rather than storing entire replicas. Alternatively, the use of some kind of homomorphic encryption might radically change the architecture.

Granularity

Although sources generate recommendations, the effective unit of data exchange within GRAFT is the profile. Further research is required to determine if using recommendations as the most granular unit would be more efficient, or provide benefits that are not possible when using profiles as the most granular unit (for example, the avoidance of concurrency issues). In particular, it might be possible to use some form of indirection to more efficiently distribute and store profiles within GRAFT.

Context Enumeration

While GRAFT supports multiple contexts, no research has been undertaken to identify and classify all of the possible contexts. This is useful research because without it, source and consumer entities will not be able to recognise and compare contexts. In particular, a classification of contexts will support the exchange of recommendations between different entities, but also the generalisation of recommendations. The latter allows for the decomposition of a recommendation generated in a specific context to a more generic context. For example, a recommendation generated in the “auctioneering” context might be generalised to the more general “buying and selling” context.

7.3.3 DHT

Attacks

The GRAFT framework utilises a distributed hash table. It was noted in Chapter 3 that this leaves GRAFT open to both Sybil and Eclipse attacks. Research is required to evaluate possible mitigations against these attacks, and how they affect the network and its performance. In particular, as GRAFT can carry reputation information about electronic entities, being able to utilise this information during the normal operation of the network may help to counter some attacks. The proposed defence would utilise the reputation information carried by GRAFT during the creation of new nodes. The number of new nodes that an entity would be allowed to create would be proportional to their reputation. As a component of this work, the ability to map a node ID to an OpenID will need to be considered. Further, as GRAFT may have a number of long-lived nodes, using these to moderate new node registrations may also help to play an important role in the mitigation of Sybil attacks.

Concurrency

The current architecture does not explicitly handle concurrency issues. An optimistic mechanism seems most appropriate for GRAFT. However, further work is required to determine how a node might determine that an update was in progress and abort its own update.

Protocol

This thesis makes extensive use of key/value pair storage within the Kademlia distributed hash table for storing profiles. Further research is required on minor protocol changes that would make the fetching and storing of profiles more efficient. In particular, and as mentioned above, reducing the granularity of the information stored within GRAFT might make for further efficiency.

Replication

The work in this thesis avoids using Kademlia replication. Further research is required to determine whether the performance of the distributed hash table could be improved by enabling replication, and how this might affect the current approach to profile verification using threshold voting.

Appendix A

Background

This chapter provides an introduction and overview to some of the key technologies that GRAFT builds upon. This chapter is not intended to provide a detailed introduction, but instead focuses on those aspects that are necessary to understanding the work in this thesis.

A.1 OpenID

OpenID [258] is a decentralised authentication protocol that allows a user to maintain a single digital identity. This identity can be registered with many sites. Each site, also known as a **Relying-Party** (RP), stores only the user's OpenID and any locally required state. The RP does not need to implement any authentication mechanisms, or store any authentication related information. Instead, authentication is only ever performed by the **OpenID Provider** (OP) where the OpenID is registered.

Figure A.1 shows a standard OpenID authentication sequence. A user wishing to obtain a service from an RP must first authenticate itself. The user presents their OpenID to the RP, and is redirected back to his or her OP. The OP authenticates the user using a previously agreed method. The actual authentication method is unspecified in the OpenID standard[1], however in most instances username and password authentication is used. Once authenticated, the user is redirected back to the RP with a cryptogrphic token generated by the OP. The token is verified by the RP by establishing a private connection with the OP. Once verified, the RP can provide the requested service to the user.

An extension to the OpenID protocol called Attribute eXchange (AX) [134] allows the relying party to request a set of attributes that further describe the authenticating user. These attributes usually include information such as the full name, email address and gender of the user. However, an OP may refuse to provide any or all of the requested

[1]`http://openid.net/specs/openid-authentication-2_0.html` - last accessed October 2014.

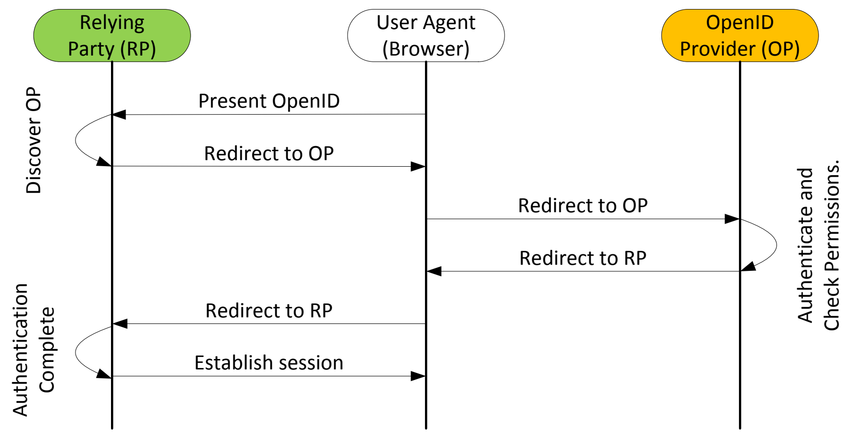

Figure A.1: OpenID Authentication. A standard OpenID authentication sequence. A user wishing to obtain a service from an RP must first authenticate itself. The user presents their OpenID to the RP, and is redirected back to his or her OP. The OP authenticates the user using a previously agreed method.

attributes, depending on the user's privacy preferences.

A.1.1 OpenID Discovery

In order for the RP to automatically find the OP during authentication, it uses something called endpoint discovery. Endpoint discovery allows the RP to find a URL where it can obtain service from the OP.

The exact type of endpoint discovery used depends on the type of OpenID used. The two possible types are URL and XRI-based, however only URL-based OpenID will be discussed. The OpenID specification states that endpoint discovery will be attempted using two mechanisms, the Yadis Protocol and the HTML-Discovery Protocol. The Yadis Protocol is always attempted first, followed by the HTML-Discovery Protocol if that fails. By preference, the discovery protocol should yield a service descriptor (in the form of an XRDS file), or simply a URL.

XRDS document

XRDS is an XML-based file format that allows for the discovery of metadata about a resource. The XRDS document often resides at a web location associated with the resource. An XRDS often contains a number of service descriptors, each which describe a service type and a location where this service can be found. Listing A.1 shows an example XRDS document that contains two service descriptors. The first describes an

OpenID service (http://specs.openid.net/auth/2.0/signon), while the second describes an OpenRep service (http://specs.open-rep.net/rx/1.0/).

```
<?xml version="1.0" encoding="UTF-8"?>
<xrds:XRDS xmlns:xrds="xri://\$xrds" xmlns="xri://\$xrd*(\$v*2.0)">
  <XRD>
    <Service priority="10">
     <Type>http://specs.openid.net/auth/2.0/signon</Type>
     <URI>https://provider.org/endpoint/</URI>
    </Service>
    <Service priority="50">
      <Type>http://specs.open-rep.net/rx/1.0/</Type>
      <URI>http://provider.com/endpoint/</URI>
    </Service>
  </XRD>
</xrds:XRDS>
```

Listing A.1: Example of an XRDS document

The Yadis Protocol

The Yadis protocol[2] attempts to obtain one or more relevant service descriptors from an XRDS document. In many instances the OpenID will itself point directly at an XRDS document. The XRDS document will contain one or more service descriptors that tell the discoverer where to find services for that OpenID. The example service descriptor in Listing A.2 shows an OpenID2 endpoint. The URI field tells the discoverer where the OP for the current endpoint can be found.

```
<Service priority="40">
  <Type>http://specs.openid.net/auth/2.0/signon</Type>
  <URI>https://provider.org/endpoint/</URI>
</Service>
```

Listing A.2: Example of an XRDS service descriptor

In some instances the OpenID may point to a HTML document. In these cases, an *X-XRDS-Location* meta tag in the HTML header may be provided. The URL within this meta tag, if it exists, will point to an XRDS document. Listing A.3 shows an example of such a meta tag.

```
<meta http-equiv="X-XRDS-Location" content="http://forum.com/?q=xrds" />
```

Listing A.3: Example of a HTML meta tag

[2]`http://yadis.org/` - last accessed January 2013.

HTML-Discovery Protocol

If the Yadis Protocol failed to resolve to a meaningful XRDS document, or the XRDS document did not contain any relevant service descriptors, the HTML-Discovery protocol will be used. This protocol requires the discoverer to search for OpenID2 link tags within the HTML header of the document found at the endpoint. Listing A.4 shows the URL of an OpenID2 endpoint within a HTML link tag.

```
<link rel="openid2.provider" href="http://provider.org/endpoint/" />
```

Listing A.4: Example of a HTML link tag

A.2 OpenRep

OpenRep is the collective name for the GRAFT OpenID extensions. OpenRep has three major components:

- OpenRep discovery
- OpenRep authentication transfer
- OpenRep web interfaces

These components are described in the following sections.

A.2.1 OpenRep Discovery

As with OpenID, a GP can be discovered using OpenID endpoint discovery. The URL discovered in an XRDS document or in the HTML document at the endpoint will point to a GRAFT provider. This provider will maintain the entity's "default" profile copy. In Listing A.5, the target has made available an XRDS document at the OpenID Identifier endpoint. The XRDS document has one service entry for the target's provider.

```
<Service priority="50">
  <Type>http://specs.open-rep.net/rx/1.0/</Type>
  <URI>http://provider.com/endpoint/</URI>
</Service>
```

Listing A.5: Service descriptor for GRAFT provider

In those cases where an OpenID endpoint does not have an XRDS document, HTML discovery may yield appropriate meta tags in the HTML document. Listing A.6 shows the same provider link as the previous example.

```
<link rel="openrep.provider" href="http://provider.com/endpoint/" />
```

Listing A.6: HTML link tag for GRAFT provider

A.2.2 OpenRep Authentication Transfer

As discussed in Section 3.3, the last step of the OpenID authentication sequence includes a profile transfer from the GP. This copy is obtained using the Attribute eXchange (AX) extension to OpenID. This extension is widely supported and is often used to transfer nickname or email address details to the RP. In this instance the AX extension is used to transfer a profile with a single recommendation. The recommendation is the one generated by the RP, or one from another site, but in the same context.

A.2.3 OpenRep Web Interfaces

Once located, the GP maintains a RESTful interface [109] implemented using HTTP. Aside from the standard OpenID functions, each GP is able to fetch and store profiles, using the target entity's OpenID as a key. The profile for a given entity can be retrieved using HTTP GET with the following URL syntax:

```
http://provider.com/?openrep.ns=http://specs.open-rep.net/rx/1.0/&openrep.mode=
    fetch&openrep.identity=openid
```

Listing A.7: OpenRep HTTP Fetch

If the resulting HTTP status code is 200, the response body will contain a valid profile. Otherwise, the response body will contain an error message. Profiles can be updated using HTTP PUT with the following URL syntax:

```
http://provider.com/?openrep.ns=http://specs.open-rep.net/rx/1.0/&openrep.mode=
    store&openrep.identity=openid
```

Listing A.8: OpenRep HTTP Store

The profile for the user should be in plain ASCII and contained in the body of the PUT operation. If the resulting HTTP status code is 200, the profile was stored correctly. Otherwise, the response body will contain an error message. Providers may reject profiles where the "updated" element is earlier than the one it already has.

A.3 Distributed Hash Tables

A distributed hash table (DHT) is a distributed approach to the classic hash table. Keys and their associated values (often called key/value pairs) are stored in the nodes that comprise the system. Any participating node can retrieve a value using only its key. In theory, a DHT can locate any given key within $O(\log N)$ hops on average, where N is the total number of peers [198].

Nodes joining the system are assigned an ID, often randomly generated or based on their IP address [298, 247]. In a DHT that is structured as a ring of N nodes, each node will then get responsibility for $1/N$ of the total key space. When a new key/value pair is introduced into the system, it is routed towards the nodes whose ID are most similar to the key. In many cases, the key is simply a hash of the value that is being stored. SHA1 is used as the hashing algorithm in both Chord [298] and Kademlia [211]. Replicas of values could be maintained in a distributed hash table by associating multiple hashes with a single value.

In general, the mapping of keys to nodes is done in such a way as to minimise the impact on the system of any nodes joining or leaving. Consistent Hashing [169] is one such approach. Consistent hashing works by assigning keys to nodes in such as way that when a node joins or leaves, it only affects a small number of keys. In theory, only K/N keys are affected, where K is the total number of keys and N is the total number of peers [298].

As a direct result of both the $O(\log N)$ key lookup performance and consistent hashing, DHTs are able to scale to large numbers of nodes and perform well even under churn.

A.3.1 Kademlia

Kademlia is a distributed hash table implementation that was designed by Maymounkov and Mazières [211].

Nodes joining the network are assigned a 160-bit identifier, generated using the SHA-1 hash function across a random number [103]. This same identifier is also used by the Kademlia protocol to locate keys that are stored in the network. The distance between any two nodes in Kademlia is based on the exclusive OR (XOR) operation. Nodes in the network are structured into a binary tree, where every node is effectively a leaf in the tree. The shortest unique prefix of the node's identifier controls a node's location in the tree. The XOR distance metric means that the distance from node x to y is the same as the distance from node y to x. This last part is an important property of Kademlia as other DHTs need to implement a stabilization protocol in order to compensate for the asymmetric nature of the overlay [41].

Appendix B

Comparison of GRAFT and XACML

B.1 Comparison of GRAFT and XACML

The XML Access Control Mark-up Language (XACML) [123] is "standard that describes both a policy language and an access control decision request/response language"[1].

XACML is primarily an Attribute-Based Access Control system (ABAC), where all access decisions are based on characteristics of three elements: the requestor, the resource being requested and the environment. The characteristics of any of these elements are called attributes, and are used during policy evaluation to decide if the requestor should have access.

The policy language in XACML is used to define access control requirements, while the request/response language allows a policy enforcement point to query if an action should be allowed and understand the result of that query.

XACML operates using the three attribute elements identified above, plus an additional "action" attribute element:

- **Subject Attributes**. A subject represents an entity such as a user or electronic service [348, 252, 123]. Every subject has a set of attributes that help to identify and define the subject. Common attributes for subjects include name, title, role and organisation.

- **Resource Attributes**. A resource, sometimes called an object, has an action performed on it by a subject. Resources include information, services or hardware. Common attributes for resources include name, owner, creation and/or last changed dates. Resource attributes are sometimes extracted from the metadata associated with the resource [348, 252], for example the dimension metadata might be extracted from an image resource.

[1] `https://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html` - last accessed October 2014.

- **Action Attributes**. An action is an operation on a resource [123].
- **Environment Attributes**. Environmental attributes are those that define the context of the environment in which the subject and resource reside. These attributes are independent of the subject, resource or action [123]. Typical examples include the time and date, operational state and resource availability.

While GRAFT is geared towards providing up to date attributes for both subjects and resources, it does not explicitly consider separate action or environmental attributes. These types of information are typically handled via variables in the policy language. Determining which policies apply when using XACML is handled using dynamic binding. In GRAFT, it is the situation that determines which policies are executed. If the user is seeking access to a particular resource, it is the attributes of the subject and resource, along with the access request that cause a GRAFT client to evaluate a particular policy. This makes it simple to determine which policy is evaluated when.

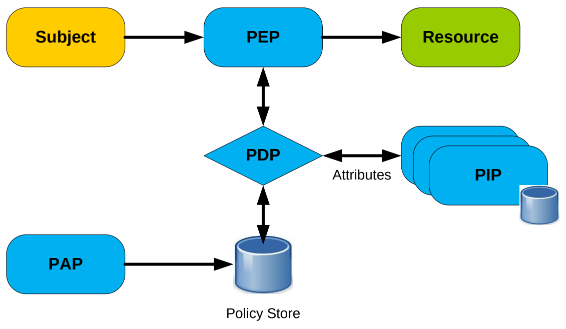

Figure B.1: A simple attribute-based authorisation architecture.

Figure B.1 shows a simple attribute-based authorisation architecture. Using the terminology promogulated by [314] (with the exception of the Policy Administration Point), the key components of this architecture can be described as:

- **Policy Enforcement Point**. The PEP intercepts the subject's request to access the resource, and asks the PDP to make a decision about the request. The PEP acts on the decision received from the PDP by either allowing or denying the request.
- **Policy Decision Point**. The PDP is responsible for evaluating the stored policies against access requests received from the PEP [252]. Any subject or resource attributes required during policy evaluation are obtained from the PIPs.
- **Policy Information Point**. PIPs are the source of subject, resource and environmental attribute values. Although a PIP is not necessarily the originator of these

attribute values, it is nevertheless responsible for obtaining the attributes and ensuring that they are matched to the right subjects, resources and environments.

- **Policy Administration Point**. The PAP manages the stored policies. Policies typically consist of rules for accessing resources. The rules may consist of conditional statements involving attributes related to the subject, resource, action or environment.

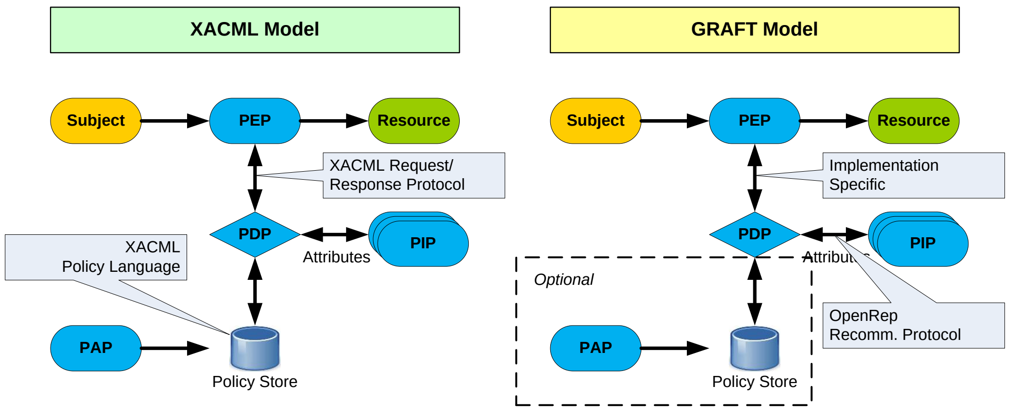

Figure B.2: Comparison of XACML and GRAFT.

In XACML, the policy language is used to define access control rules and conditions. Many rules can be combined into one policy, which in turn can be combined to form a policy set. XACML depends on the PAP to author appropriate policies and on the PIPs to create and maintain attributes for subjects, resources and environments. These policies and attributes are often created manually. The potential number of attributes on subjects and resources (the NSA agreed to 13 subject attributes for use within the US Department of Defense [171]), can lead to significant work.

In GRAFT however, the PAP is optional, as it is up to each individual PDP to decide how it will manage its access control policies. The examples in this paper have used a PHP-based rules engine, however a PDP could equally have implemented its policies using XACML. GRAFT sources can be seen as dynamic PIPs, collecting and making available attributes for subjects and resources. As mentioned previously, GRAFT does not handle attributes for actions and environments, these are handled as part of the access control decision.

Likewise, in GRAFT, there is no explicit requirement for a separate access control request/response language as each policy decision point can implement their access control policies as they desire. In fact, the XACML Request/Response language could be utilised if desired.

Appendix C

SimGrid

A simulation using the SimGrid [58] simulation software was built initially. The experimental work using this simulation software was stopped at the base case because of unexpected and unexplainable results. This section documents some of those results.

C.1 SimGrid

SimGrid[1] is a tool designed to study large-scale, distributed systems. SimGrid provides a library of routines that a simulation is linked against, giving us an API for the control of processes and messages.

All SimGrid simulations require a platform file that describes the networks and hosts that simulated peers will run on. All of the experiments were based on two platform files. The first, called "Cluster", described a single large cluster that contained all of the hosts. The second, called "Grid5000", described a modified Grid'5000 environment[2]. The structure of the autonomous systems (AS) and the routing between them was not changed for the experiments. However, clusters within each AS were modified to hold the same number of hosts, so that a uniform distribution of hosts was possible.

Similarly, every SimGrid simulation also requires a deployment file that controls how the simulated peers are deployed across the hosts specified in the platform file. In each simulation, peers are deployed to hosts either randomly, or in a uniform fashion such that clusters are filled evenly.

The simulation, based in part on sample code provided as part of the SimGrid distribution, implements a Kademlia DHT with n identical peers and a single service composer peer. Each peer joins the network, and then answers "find node" and "profile queries". The composer peer waits until after the network has stabilised, and then issues profile queries to p peers for each service s that it is interested in, resulting in a total of q profile

[1] `http://simgrid.gforge.inria.fr/` - last accessed April 2014.

[2] `https://www.grid5000.fr/` - last accessed April 2014.

queries being sent. The number of services s and peers p is varied as required. The p peers for each run are randomly selected from the available peers before the simulation.

A simple threshold was implemented so that although p peers are queried, the lookup of a profile was considered complete when a threshold of t profile query responses is in agreement. The threshold value is set to $^3/_4$ of the queried peers. For example, if there are 11 peers, the threshold value will be 8 profile query responses.

C.1.1 Cluster Platform

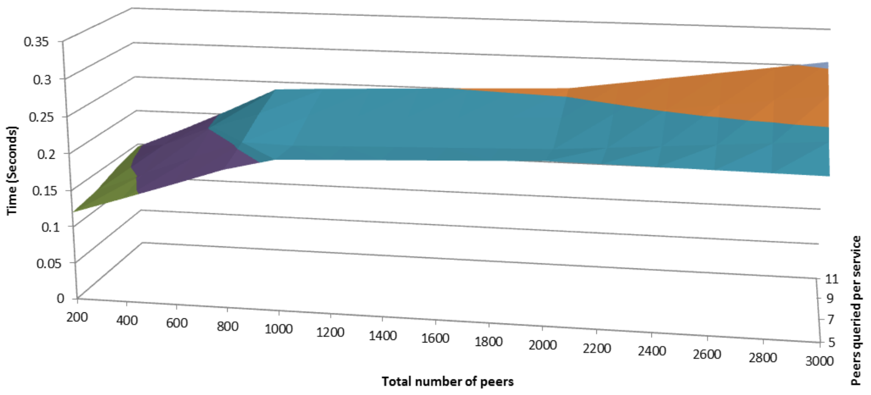

Figure C.1: Total time for $s = 3$ on Cluster Platform.

As can be seen from Figure C.1, the average total time to receive t profiles increased linearly with the number of peers queried p for each service and with the total number of peers n. In contrast to the graphs presented in Chapter 6, the slope on the graph changes noticeably at two points. The first of these is at around 800 nodes, and the second at around 2000 nodes.

The inter-arrival time for profiles is shown in Figure C.2. This graph shows a trend toward decreasing inter-arrival times with increasing queries q, however the slope changes at 800 and 2000 nodes are again visible. There is nothing apparent in the configuration or simulated environment that would allow an explanation for these inflexion points.

C.1.2 Grid5000 Platform

The average total time to receive t profiles is shown in Figure C.3. This figure shows the same linear increase in time as Figure C.1, however there is an unexpected increase in time at 800 nodes, and another change in trend at 2000 nodes. The increase at 800 nodes represents approximately 0.3 seconds.

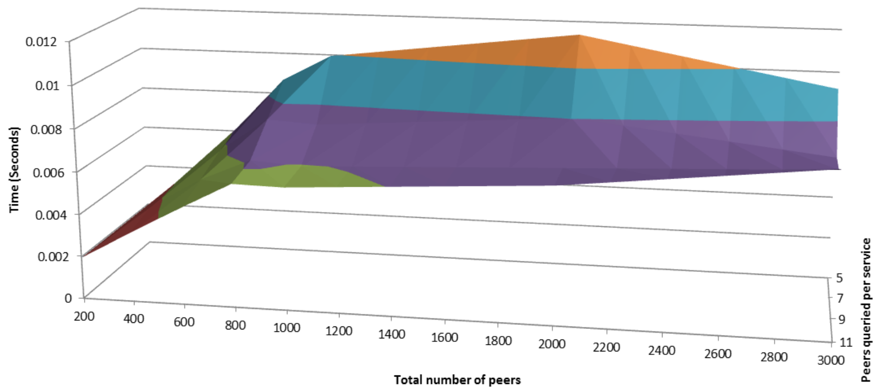

Figure C.2: Response inter-arrival time for $s = 3$ on Cluster Platform.

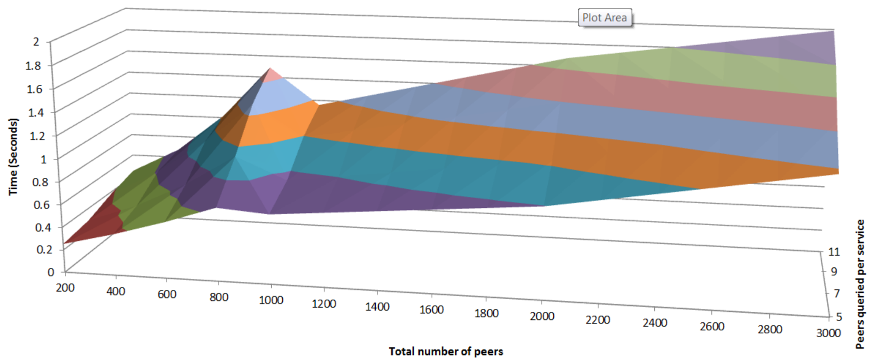

Figure C.3: Total time for $s = 3$ on Grid'5000 Platform.

Given that the entire platform and deployment model for the nodes in this simulation have changed, the issues at 800 and 2000 nodes are unexplained from a simulation perspective. A reasonable, but unproven, explanation for these issues is that the simulator implementation has a number of internal limits, and once these are reached, it changes the overall time taken to process the simulation.

Appendix D

Samples

This chapter provides sample listings for some of the key components developed in Chapter 5: Implementation.

D.1 GRAFT Profiles

The fictional profile given in Listing D.1 contains two recommendations. The subject, or target, of both recommendations is Alice. The first recommendation, from a forum site, describes a simple membership relationship, with a rating score of 126. This recommendation may be useful to other sites that operate within the same context, which in this instance is enumerated as "public.forum". The second recommendation, from a fictional DBLP source evaluates Alice in an academic context. This recommendation source has assigned Alice a H-index score of 31.

```xml
<?xml version="1.0"?>
<container xmlns="http://specs.open-rep.net/rx/1.0/">
  <recommendations count="2">
    <recommendation id="http://www.forum.net/">
      <target type="openid">http://id.example.net/alice/</target>
      <source type="openid">http://www.forum.net/</source>
      <assert type="member" context="public.forum"/>
      <assert type="rating" context="public.forum">126</assert>
      <updated>2013-07-16T03:23:45+00:00</updated>
    </recommendation>
    <recommendation id="http://scholar.dblp.org/">
      <target type="openid">http://id.example.net/alice/</target>
      <source type="openid">http://scholar.dblp.org/</source>
      <assert type="hindex" context="academic.scholarship">31</assert>
      <updated>2013-07-18T08:11:59+00:00</updated>
    </recommendation>
  </recommendations>
```

```
</container>
```

Listing D.1: Sample of a profile

In the sample profile in listing D.2, Alice has nominated Bob as her delegate in the "public.forum" context. This means that Bob may temporarily utilise Alice's permissions in that given context. If Alice does not have any permissions in the system in question, then Bob is not able to obtain them through Alice.

```
<?xml version="1.0"?>
<container xmlns="http://specs.open-rep.net/rx/1.0/">
  <recommendations count="1">
    <recommendation id="http://provider.com/endpoint/">
      <target type="openid">http://id.example.net/alice/</target>
      <source type="openid">http://provider.com/endpoint/</source>
      <delegate context="public.forum" type="openid">http://id.example.net/bob/
          </delegate>
      <updated>2013-07-19T12:51:10+00:00</updated>
    </recommendation>
  </recommendations>
</container>
```

Listing D.2: Sample of a profile with delegation

D.2 GRAFT Policies

In the implementation described in Chapter 5, GRAFT polices were written using the PHP-based rules engine Ruler. The following sample policies were utilised in the implementation. Equations 5.1 and 5.2 describe common situations in the forum scenario described in Section 5.1, and are implemented in Ruler by the following code fragments.

```
$rb->logicalOr(
    $rb['rating']->lessThan(10),
    $rb['admin']->equalTo(null)
)
```

Listing D.3: Sample Ruler policy for Equation 5.1

```
$rb->logicalAnd(
    $rb['rating']->greaterThanOrEqualTo(100),
    $rb['forum']->notEqualTo('administration')
)
```

Listing D.4: Sample Ruler policy for Equation 5.2

In a similar fashion, Section 5.2 describes the implementation of a GRAFT-enabled wiki. Equations 5.3 and 5.4 describe common situations in a wiki scenario, and are implemented in Ruler by the following code fragments.

```
$rb->logicalAnd(
    $rb['hindex']->greaterThanOrEqualTo(1),
    $rb->logicalAnd(
        $rb['degree']->greaterThanOrEqualTo(0),
        $rb['degree']->lessThanOrEqualTo(3)
    )
)
```

Listing D.5: Sample Ruler policy for Equation 5.3

```
$rb->logicalAnd(
    $rb['hindex']->greaterThanOrEqualTo(20),
    $rb->logicalAnd(
        $rb['degree']->greaterThanOrEqualTo(0),
        $rb['degree']->lessThanOrEqualTo(2)
    )
)
```

Listing D.6: Sample Ruler policy for Equation 5.4

D.3 Workflow

The Anduril workflow given in Listing D.7 illustrates the use of the "CallService" component that was written for GRAFT. This component calls one or more web services with the given input data. The first parameter passed to the component is a list of URLs that represent web services. These are tried in order, until one succeeds. Any data pased to the component as the second parameter is passed to the web service. Data returned from the call is made available to the workflow engine via the output of the component.

```
data = INPUT(path="data.txt")
list = INPUT(path="list.csv")

output = CallService(list, data)

OUTPUT(output)
```

Listing D.7: Sample workflow with CallService

The Anduril workflow in Listing D.8 is almost identical to the previous listing, with the exception that the "RankService" component is initially called to generate a list of web services. This component is passed only a single parameter, the "category", that controls the context of the web services that are returned.

```
data = INPUT(path="data.txt")
list = RankService(category="public.services")

output = CallService(list, data)
```

```
OUTPUT(list)
OUTPUT(output)
```

Listing D.8: Sample workflow with RankService

D.4 Oversim

The OverSim configuration given in Listing D.9 was used for the churn experiments discussed in Section 6.9.

```
[Config GRAFT]
description = GRAFT
repeat = 40
seed-set = ${runnumber}
**.transitionTime = 500s
**.measurementTime = 9000s
**.churnGeneratorTypes = "oversim.common.NoChurn␣oversim.common.ParetoChurn"
**.churnGenerator[0].targetOverlayTerminalNum = 1
**-0[*].overlayType = "oversim.overlay.kademlia.KademliaModules"
**-0[*].tier1Type = "oversim.applications.graft.GCompModules"
**.churnGenerator[1].targetOverlayTerminalNum = ${N=1000, 2000, 3000, 4000,
    5000, 6000, 7000, 8000, 9000}
**-1[*].overlayType = "oversim.overlay.kademlia.KademliaModules"
**-1[*].tier1Type = "oversim.applications.graft.GNodeModules"
**-1[*].lifetimeMean = 100s
**-1[*].deadtimeMean = 100s
**.initPhaseCreationInterval = 0.01s
**.sampPeriod = 1000s
**.waitPeriod = 8000s
**.numServices = ${S=3, 5}
**.numPeers = ${P=5, 7, 9, 11}
**.numQueries = ${Q=($S)*($P)}
**.overlay.kademlia.lookupRedundantNodes = 16
**.overlay.kademlia.s = 8
**.overlay.kademlia.k = 16
**.overlay.kademlia.lookupMerge = true
**.overlay.kademlia.lookupParallelPaths = 1
**.overlay.kademlia.lookupParallelRpcs = 1
```

Listing D.9: Sample OverSim Configuration

Bibliography

[1] Advogato. http://www.advogato.org/. Last accessed 2014-04-07.

[2] Amazon. http://www.amazon.com/. Last accessed 2014-04-07.

[3] Couchsurfing. http://www.couchsurfing.org/. Last accessed 2014-04-07.

[4] Digg. http://digg.com/. Last accessed 2014-04-07.

[5] Dropbox. http://www.dropbox.com/. Last accessed 2014-04-07.

[6] eBay. http://www.ebay.com/. Last accessed 2014-04-07.

[7] ePinions. http://www.epinions.com/. Last accessed 2014-04-07.

[8] Facebook. http://www.facebook.com/. Last accessed 2014-04-07.

[9] Google. http://www.google.com/. Last accessed 2014-04-07.

[10] LinkedIn. http://www.linkedin.com/. Last accessed 2014-04-07.

[11] MTurk. http://www.mturk.com/. Last accessed 2014-04-07.

[12] Reddit. http://www.reddit.com/. Last accessed 2014-04-07.

[13] Slashdot. http://slashdot.org/. Last accessed 2014-04-07.

[14] Stackoverflow. http://stackoverflow.com/. Last accessed 2014-04-07.

[15] TrustedSource. http://www.trustedsource.org/. Last accessed 2014-04-07.

[16] Trustribe. http://www.trustribe.com/. Last accessed 2014-04-07.

[17] Turkopticon. http://turkopticon.differenceengines.com/. Last accessed 2014-04-07.

[18] Wikipedia. http://www.wikipedia.org/. Last accessed 2014-04-07.

[19] Yelp. http://www.yelp.com/. Last accessed 2014-04-07.

[20] ABDUL-RAHMAN, A., AND HAILES, S. A distributed trust model. In *Proceedings of the 1997 workshop on New security paradigms* (1998), ACM, pp. 48–60.

[21] ABDUL-RAHMAN, A., AND HAILES, S. Supporting trust in virtual communities. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on* (2000), IEEE, pp. 9–pp.

[22] ABERER, K. P-Grid: A Self-Organizing Access Structure for P2P Information Systems. In *Cooperative Information Systems*, C. Batini, F. Giunchiglia, P. Giorgini, and M. Mecella, Eds., vol. 2172 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2001, pp. 179–194.

[23] ABERER, K., CUDRÉ-MAUROUX, P., DATTA, A., DESPOTOVIC, Z., HAUSWIRTH, M., PUNCEVA, M., AND SCHMIDT, R. P-Grid: A Self-Organizing Structured P2P System. *SIGMOD Rec. 32* (September 2003), 29–33.

[24] ABERER, K., AND DESPOTOVIC, Z. Managing Trust in a Peer-2-Peer Information System. In *Proceedings of International Conference on Information Knowledge Management* (2001), CIKM, ACM.

[25] ABERER, K., AND DESPOTOVIC, Z. Managing trust in a peer-2-peer information system. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management* (New York, NY, USA, 2001), ACM, pp. 310–317.

[26] AGUDO, I., FERNANDEZ-GAGO, C., AND LOPEZ, J. A Multidimensional Reputation Scheme for Identity Federations. In *Public Key Infrastructures, Services and Applications*, F. Martinelli and B. Preneel, Eds., vol. 6391 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 225–238.

[27] AGUDO, I., FERNANDEZ-GAGO, C., AND LOPEZ, J. A multidimensional reputation scheme for identity federations. In *Public Key Infrastructures, Services and Applications*, F. Martinelli and B. Preneel, Eds., vol. 6391 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2010, pp. 225–238.

[28] AIELLO, L. M., MILANESIO, M., RUFFO, G., AND SCHIFANELLA, R. Tempering kademlia with a robust identity based system. In *Peer-to-Peer Computing, 2008. P2P'08. Eighth International Conference on* (2008), IEEE, pp. 30–39.

[29] AKAVIPAT, R., AL-AMEEN, M., KAPADIA, A., RAHMAN, Z., SCHLEGEL, R., AND WRIGHT, M. ReDS: A Framework for Reputation-Enhanced DHTs. *Parallel and Distributed Systems, IEEE Transactions on 25*, 2 (Feb 2014), 321–331.

[30] ALBRECHT, W. S., ALBRECHT, C. O., ALBRECHT, C. C., AND ZIMBELMAN, M. F. *Fraud Examination*, 4th ed. South-Western College Publications, 2011.

[31] BAFOUTSOU, G., AND MENTZAS, G. Review and functional classification of collaborative systems. *International journal of information management 22*, 4 (2002), 281–305.

[32] BAGHERI, E., AND GHORBANI, A. A. Behavior Analysis through Reputation Propagation in a Multi-context Environment. In *PST '06: Proceedings of the 2006 International Conference on Privacy, Security and Trust* (New York, NY, USA, 2006), ACM, pp. 1–7.

[33] BAGHERI, E., ZAFARANI, R., AND BAROUNI-EBRAHIMI, M. Can reputation migrate? on the propagation of reputation in multi-context communities. *Knowledge-Based Systems 22*, 6 (2009), 410 – 420.

[34] BAILEY, K. D. *Typologies and Taxonomies: An Introduction to Classification Techniques*. Sage, 1994.

[35] BAKKER, A., AMADE, E., BALLINTIJN, G., KUZ, I., VERKAIK, P., VAN DER WIJK, I., VAN STEEN, M., AND TANENBAUM, A. S. The Globe Distribution Network. In *Proceedings of FREENIX Track: 2000 USENIX Annual Technical Conference* (June 2000).

[36] BALAKRISHNAN, H., KAASHOEK, M. F., KARGER, D., MORRIS, R., AND STOICA, I. Looking up data in P2P systems. *Communications of the ACM 46* (February 2003), 43–48.

[37] BALASUBRAMANIAN, V., AND BASHIAN, A. Document Management and Web technologies: Alice marries the Mad Hatter. *Communications of the ACM 41*, 7 (1998), 107–115.

[38] BALLINTIJN, G., VAN STEEN, M., AND TANENBAUM, A. S. Scalable naming in global middleware. In *Proceedings of the 13th International Conference on Parallel and Distributed Computing Systems* (August 2000), PDCS-2000, pp. 624–631.

[39] BASNEY, J. Credential wallets. Tech. rep., National Center for Supercomputing Applications, University of Illinois, USA, 2001.

[40] BAUMGART, I., HEEP, B., AND KRAUSE, S. OverSim: A Flexible Overlay Network Simulation Framework. In *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA* (May 2007), pp. 79–84.

[41] BAUMGART, I., AND MIES, S. S/Kademlia: A practicable approach towards secure key-based routing. In *Parallel and Distributed Systems, 2007 International Conference on* (2007), vol. 2, IEEE, pp. 1–8.

[42] BEVERLY YANG, B., AND GARCIA-MOLINA, H. Designing a super-peer network. In *Data Engineering, 2003. Proceedings. 19th International Conference on* (2003), IEEE, pp. 49–60.

[43] BHAGWAN, R., MOORE, D., SAVAGE, S., AND VOELKER, G. Replication strategies for highly available peer-to-peer storage. *Future directions in distributed computing* (2003), 153–158.

[44] BHAGWAN, R., SAVAGE, S., AND VOELKER, G. Understanding availability. *Peer-to-Peer Systems II* (2003), 256–267.

[45] BHATTI, R., BERTINO, E., AND GHAFOOR, A. A trust-based context-aware access control model for web-services. In *Web Services, 2004. Proceedings. IEEE International Conference on* (2004), IEEE, pp. 184–191.

[46] BIGHAM, J. P., AND LADNER, R. E. What the disability community can teach us about interactive crowdsourcing. *interactions 18*, 4 (2011), 78–81.

[47] BLAZE, M., FEIGENBAUM, J., AND LACY, J. Decentralized Trust Management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy* (1996), IEEE Computer Society Press, pp. 164–173.

[48] BOYD, D. M., AND ELLISON, N. B. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication 13*, 1 (2007), 210–230.

[49] BRABHAM, D. C. Crowdsourcing as a model for problem solving an introduction and cases. *Convergence: The International Journal of Research into New Media Technologies 14*, 1 (2008), 75–90.

[50] BRODER, A., MITZENMACHER, M., AND MITZENMACHER, A. B. I. M. Network applications of bloom filters: A survey. In *Internet Mathematics* (2002), pp. 636–646.

[51] BUBENDORFER, K., CHARD, K., JOHN, K., AND THAUFEEG, A. M. eScience in the Social Cloud. *Future Generation Computer Systems 29*, 8 (2013), 2143–2156.

[52] BUCHEGGER, S., AND BOUDEC, J.-Y. L. Performance analysis of the CONFIDANT protocol. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing* (2002), ACM, pp. 226–236.

[53] BUCHEGGER, S., AND BOUDEC, J. Y. L. A robust reputation system for mobile ad-hoc networks. In *Proceedings of P2PEcon* (2003).

[54] BUTLER, R., WELCH, V., ENGERT, D., FOSTER, I., TUECKE, S., VOLMER, J., AND KESSELMAN, C. A national-scale authentication infrastructure. *Computer 33*, 12 (dec 2000), 60 – 66.

[55] CALLAGHAN, S., MAECHLING, P., DEELMAN, E., VAHI, K., MEHTA, G., JUVE, G., MILNER, K., GRAVES, R., FIELD, E., OKAYA, D., ET AL. Reducing Time-to-Solution using Distributed High-Throughput Mega-Workflows - Experiences from SCEC CyberShake. In *eScience, 2008. eScience'08. IEEE Fourth International Conference on* (2008), IEEE, pp. 151–158.

[56] CAPON, N. Credit Scoring Systems: A Critical Analysis. *The Journal of Marketing* (1982), 82–91.

[57] CARMINATI, B., FERRARI, E., AND PEREGO, A. Enforcing access control in web-based social networks. *ACM Transactions on Information and System Security (TISSEC) 13*, 1 (2009), 6.

[58] CASANOVA, H., LEGRAND, A., AND QUINSON, M. SimGrid: a Generic Framework for Large-Scale Distributed Experiments. In *Proceedings of the Tenth International Conference on Computer Modeling and Simulation* (Washington, DC, USA, 2008), UKSIM '08, IEEE Computer Society, pp. 126–131.

[59] CASARE, S., AND SICHMAN, J. Towards a Functional Ontology of Reputation. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems* (New York, NY, USA, 2005), ACM, pp. 505–511.

[60] CASATI, F., ILNICKI, S., JIN, L., KRISHNAMOORTHY, V., AND SHAN, M.-C. Adaptive and Dynamic Service Composition in eFlow. In *Advanced Information Systems Engineering*, B. Wangler and L. Bergman, Eds., vol. 1789 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2000, pp. 13–31.

[61] CASCELLA, R. Enabling fast bootstrp of reputation in P2P mobile networks. In *Advanced Information Networking and Applications* (2009), AINA '09, IEEE Computer Society, pp. 371–378.

[62] CASTRO, M., DRUSCHEL, P., GANESH, A., ROWSTRON, A., AND WALLACH, D. S. Secure Routing for Structured Peer-to-Peer Overlay Networks. *ACM SIGOPS Operating Systems Review 36*, SI (2002), 299–314.

[63] CATON, S., DUKAT, C., GRENZ, T., HAAS, C., PFADENHAUER, M., AND WEINHARDT, C. Foundations of trust: Contextualising trust in social clouds. In *Cloud and Green Computing (CGC), 2012 Second International Conference on* (2012), IEEE, pp. 424–429.

[64] CATON, S., HAAS, C., CHARD, K., BUBENDORFER, K., AND RANA, O. A Social Compute Cloud: Allocating and Sharing Infrastructure Resources via Social Networks. *IEEE Transactions on Services Computing* (2014).

[65] CHADWICK, D. W. Operational models for reputation servers. In *Trust Management*, P. Herrmann, V. Issarny, and S. Shiu, Eds., vol. 3477 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005, pp. 9–23.

[66] CHAN, P. P.-W., AND LYU, M. R. Dynamic web service composition: A new approach in building reliable web service. In *Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on* (2008), IEEE, pp. 20–25.

[67] CHANG, E., HUSSAIN, F. K., AND DILLON, T. Reputation Ontology for Reputation Systems. *Lecture Notes in Computer Science 3762* (2005), 957–966.

[68] CHARD, K., BUBENDORFER, K., CATON, S., AND RANA, O. Social cloud computing: A vision for socially motivated resource sharing. *IEEE Transactions on Services Computing 5*, 4 (2012), 551–563.

[69] CHARD, K., CATON, S., RANA, O., AND BUBENDORFER, K. Social Cloud: Cloud Computing in Social Networks. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on* (2010), IEEE, pp. 99–106.

[70] CHARD, R. Reputation description and interpretation. Master's thesis, Victoria University of Wellingon, New Zealand, 2011.

[71] CHEN, B., AND ROSCOE, A. Social networks for importing and exporting security. In *Large-Scale Complex IT Systems. Development, Operation and Management*. Springer, 2012, pp. 132–147.

[72] CHEN, M., AND SINGH, J. P. Computing and using reputations for internet ratings. In *Proceedings of the 3rd ACM conference on Electronic Commerce* (New York, NY, USA, 2001), EC '01, ACM, pp. 154–162.

[73] CHEN, R., AND YEAGER, W. Poblano: A Distributed Trust Model for Peer-to-Peer Networks. Tech. rep., Sun Microsystems, 2001.

[74] CHOW, R., GOLLE, P., JAKOBSSON, M., SHI, E., STADDON, J., MASUOKA, R., AND MOLINA, J. Controlling data in the cloud: outsourcing computation without outsourcing control. In *Proceedings of the 2009 ACM workshop on Cloud computing security* (2009), ACM, pp. 85–90.

[75] CISCO SYSTEMS INC. Cisco Visual Networking Index: Forecast and Methodology, 2010-2015. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf`, June 2011. Last accessed 2014-04-07.

[76] COFFMAN, K. G., AND ODLYZKO, A. M. Internet growth: Is there a "Moore's Law" for data traffic? *Handbook of massive data sets 142* (2001).

[77] CONNER, W., IYENGAR, A., MIKALSEN, T., ROUVELLOU, I., AND NAHRSTEDT, K. A Trust Management Framework for Service-oriented Environments. In *WWW '09: Proceedings of the 18th international conference on World wide web* (New York, NY, USA, 2009), ACM, pp. 891–900.

[78] CONRAD, M., AND HOF, H.-J. A generic, self-organizing, and distributed bootstrap service for peer-to-peer networks. In *Self-Organizing Systems*, D. Hutchison and R. Katz, Eds., vol. 4725 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2007, pp. 59–72.

[79] CRAINICEANU, A., LINGA, P., MACHANAVAJJHALA, A., GEHRKE, J., AND SHANMUGASUNDARAM, J. P-Ring: an Efficient and Robust P2P Range Index Structure. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (New York, NY, USA, 2007), SIGMOD '07, ACM, pp. 223–234.

[80] CRAMER, C., KUTZNER, K., AND FUHRMANN, T. Bootstrapping locality-aware P2P networks. In *ICON 2004: 12th IEEE International Conference on Networks* (November 2004), vol. 1, pp. 357 – 361.

[81] CURTIS, N., SAFAVI-NAINI, R., AND SUSILO, W. X^2Rep: Enhanced Trust Semantics for the XRep Protocol. *Lecture Notes in Computer Science 3089* (2004), 205–219.

[82] DAMIANI, E., DI VIMERCATI, D. C., PARABOSCHI, S., SAMARATI, P., AND VIOLANTE, F. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM conference on Computer and communications security* (New York, NY, USA, 2002), CCS '02, ACM, pp. 207–216.

[83] DAMIANI, E., DI VIMERCATI, S. D. C., PARABOSCHI, S., AND SAMARATI, P. Managing and Sharing Servents' Reputations in P2P. *IEEE Transactions on Data and Knowledge Engineering 15* (2003), 840–854.

[84] DANEZIS, G., AND DIAZ, C. A survey of anonymous communication channels. *Journal of Privacy Technology* (2008).

[85] DE ROURE, D., GOBLE, C., AND STEVENS, R. The design and realisation of the myExperiment virtual research environment for social sharing of workflows. *Future Generation Computer Systems 25* (2009), 561–567.

[86] DEBATIN, B., LOVEJOY, J. P., HORN, A.-K., AND HUGHES, B. N. Facebook and online privacy: Attitudes, behaviors, and unintended consequences. *Journal of Computer-Mediated Communication 15*, 1 (2009), 83–108.

[87] DEELMAN, E., GANNON, D., SHIELDS, M., AND TAYLOR, I. Workflows and e-Science: An Overview of Workflow System Features and Capabilities. *Future Generation Computer Systems 25*, 5 (2009), 528–540.

[88] DELLAROCAS, C. The Digitization of Word of Mouth: Promise and Challenges of Online Feedback Mechanisms. *Management Science 49* (2003), 1407–1424.

[89] DESPOTOVIC, Z., AND ABERER, K. Possibilities for Managing Trust in P2P Networks. Tech. rep., Swiss Federal Institute of Technology, Zurich, Switzerland, 2004.

[90] DEWAN, P., AND DASGUPTA, P. PRIDE: Peer-to-Peer Reputation Infrastructure for Decentralized Environments. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters* (New York, NY, USA, 2004), ACM, pp. 480–481.

[91] DEY, A. K., AND ABOWD, G. D. Towards a Better Understanding of Context and Context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing* (1999), Springer-Verlag, pp. 304–307.

[92] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Reputation in privacy enhancing technologies. In *Proceedings of the 12th annual conference on Computers, freedom and privacy* (2002), ACM, pp. 1–6.

[93] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Reputation in P2P anonymity systems. In *Proceedings of Workshop on Economics of Peer-to-Peer Systems, June* (2003), Citeseer.

[94] DOAN, A., RAMAKRISHNAN, R., AND HALEVY, A. Y. Crowdsourcing systems on the world-wide web. *Commun. ACM 54* (April 2011), 86–96.

[95] DONAHOE, J. eBay annual report 2010. `http://investor.ebayinc.com/common/dar/dar.cfm?DocumentID=2932&CompanyID=ebay&zid=576ea96c`, January 2011. Last accessed 2014-04-07.

[96] DOUCEUR, J. The sybil attack. *Peer-to-peer Systems* (2002), 251–260.

[97] DOWLATSHAHI, M., MACLARTY, G., AND FRY, M. A scalable and efficient architecture for service discovery. In *The 11th IEEE International Conference on Networks* (September 2003), ICON2003, pp. 51 – 56.

[98] DUH, R. R., JAMAL, K., AND SUNDER, S. Control and assurance in e-commerce: privacy, integrity and security at eBay. *Taiwan Accounting Review 3*, 1 (2002), 1–27.

[99] DURAND, J. Experimental OpenID Service for DOEGrids. `http://www.doegrids.org/OpenID/OpenID%20Presentation-1.ppt`, June 2008. Last accessed 2014-08-17.

[100] DUSTDAR, S., AND SCHREINER, W. A survey on web services composition. *International Journal of Web and Grid Services 1*, 1 (2005), 1–30.

[101] DUTTA, D., GOEL, A., GOVINDAN, R., AND ZHANG, H. The Design of A Distributed Rating Scheme for Peer-to-Peer Systems. In *The 1st Workshop on Economics of Peer-to-Peer Systems* (2003).

[102] FÄHNRICH, S., OBREITER, P., AND KÖNIG-RIES, B. The buddy system: A distributed reputation system based on social structure. Tech. rep., 2004.

[103] FANTACCI, R., MACCARI, L., ROSI, M., CHISCI, L., AIELLO, L. M., AND MILANESIO, M. Avoiding eclipse attacks on kad/kademlia: an identity based approach. In *Proceedings of the 2009 IEEE international conference on Communications* (2009), IEEE Press, pp. 983–987.

[104] FARMER, F. R., AND GLASS, B. *Building Web Reputation Systems*, 1st ed. O'Reilly Media, Inc., 2010.

[105] FEDOTOVA, N., AND VELTRI, L. Reputation management algorithms for DHT-based peer-to-peer environment. *Computer Communications 32*, 12 (2009), 1400 – 1409. Special Issue of Computer Communications on Heterogeneous Networking for Quality, Reliability, Security, and Robustness Part II.

[106] FELSTINER, A. Working the crowd: Employment and labor law in the crowdsourcing industry. *Berkeley J. Emp. & Lab. L. 32* (2011), 143–143.

[107] FERRAIOLO, D., CUGINI, J., AND KUHN, D. R. Role-based access control (RBAC): Features and motivations. In *Proceedings of 11th Annual Computer Security Application Conference* (1995), sn, pp. 241–48.

[108] FERREIRA, W. Crowdsourcing @ IBM. Presentation given at CrowdNet 2012: 2nd Workshop on Cloud Labor and Human Computation, January 2012.

[109] FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, 2000.

[110] FOSTER, I. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Lecture Notes in Computer Science* (2001), 1–4.

[111] FRANKLIN, M., KOSSMANN, D., KRASKA, T., RAMESH, S., AND XIN, R. CrowdDB answering queries with crowdsourcing. *Proceedings of SIGMOD 2011* (2011), 61–72.

[112] FRIEDMAN, E., AND RESNICK, P. The Social Cost of Cheap Pseudonyms. *Journal of Economics and Management Strategy 10* (2001), 173–199.

[113] GAL-OZ, N., GRINSHPOUN, T., AND GUDES, E. Privacy issues with sharing and computing reputation across communities. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications 1*, 4 (2010), 16–34.

[114] GAL-OZ, N., GRINSHPOUN, T., AND GUDES, E. Privacy Issues with Sharing Reputation across Virtual Communities. In *Proceedings of the 4th International Workshop on Privacy and Anonymity in the Information Society* (2011), ACM, p. 3.

[115] GAL-OZ, N., GRINSHPOUN, T., GUDES, E., AND MEISELS, A. Cross-community reputation: Policies and alternatives. In *Proceedings of the IADIS International Conference on Web Based Communities, Amsterdam, The Netherlands* (2008), pp. 197–201.

[116] GAL-OZ, N., GUDES, E., AND HENDLER, D. A Robust and Knot-Aware Trust-Based Reputation Model. In *Trust Management II*, Y. Karabulut, J. Mitchell, P. Herrmann, and C. Jensen, Eds., vol. 263 of *IFIP The International Federation for Information Processing*. Springer US, 2008, pp. 167–182.

[117] GANNON, D., PLALE, B., MARRU, S., KANDASWAMY, G., SIMMHAN, Y., AND SHIRASUNA, S. *Workflows for e-Science: Scientific Workflows for Grids*. Springer, 2006, ch. Dynamic, Adaptive Workflows for Mesoscale Meteorology.

[118] GARG, A., MONTRESOR, A., AND BATTITI, R. Reputation lending for virtual communities. *22nd International Conference on Data Engineering Workshops 0* (2006), 22.

[119] GEIGER, D., SEEDORF, S., AND SCHADER, M. Managing the crowd: Towards a taxonomy of crowdsourcing processes. In *Proceedings of the Seventeenth Americas Conference on Information Systems* (2011).

[120] GHAFFARINEJAD, A., AND AKBARI, M. K. An incentive compatible and distributed reputation mechanism based on context similarity for service oriented systems. *Future Generation Computer Systems 29*, 3 (2013), 863 – 875. Special Section: Recent Developments in High Performance Computing and Security.

[121] GILES, C. L., BOLLACKER, K. D., AND LAWRENCE, S. Citeseer: an automatic citation indexing system. In *International Conference on Digital Libraries* (1998), ACM Press, pp. 89–98.

[122] GIROIRE, F., MONTEIRO, J., AND PERENNES, S. P2P storage systems: How much locality can they tolerate? In *IEEE 34th Conference on Local Computer Networks, 2009* (October 2009), LCN 2009, pp. 320 – 323.

[123] GODIK, S., ANDERSON, A., PARDUCCI, B., HUMENN, P., AND VAJJHALA, S. OASIS eXtensible Access Control Markup Language (XACML). Tech. rep., Tech. rep., OASIS, 2002.

[124] GOECKS, J., AND MYNATT, E. Enabling privacy management in ubiquitous computing environments through trust and reputation systems. In *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work, New Orleans, LA, USA* (2002).

[125] GOYAL, V., PANDEY, O., SAHAI, A., AND WATERS, B. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security* (2006), ACM, pp. 89–98.

[126] GRAVES, R., JORDAN, T. H., CALLAGHAN, S., DEELMAN, E., FIELD, E., JUVE, G., KESSELMAN, C., MAECHLING, P., MEHTA, G., MILNER, K., ET AL. CyberShake: A physics-based seismic hazard model for southern California. *Pure and Applied Geophysics 168*, 3-4 (2011), 367–381.

[127] GRINSHPOUN, T., GAL-OZ, N., MEISELS, A., AND GUDES, E. CCR: A model for sharing reputation knowledge across virtual communities. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01* (Washington, DC, USA, 2009), WI-IAT '09, IEEE Computer Society, pp. 34–41.

[128] GUDES, E., GAL-OZ, N., AND GRUBSHTEIN, A. Methods for Computing Trust and Reputation While Preserving Privacy. In *Data and Applications Security XXIII*. Springer, 2009, pp. 291–298.

[129] GUDES, E., GAL-OZ, N., AND GRUBSHTEIN, A. Methods for computing trust and reputation while preserving privacy. *Data and Applications Security XXIII* (2009), 291–298.

[130] GUPTA, M., JUDGE, P., AND AMMAR, M. A Reputation System for Peer-to-Peer Networks. In *NOSSDAV '03: Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video* (New York, NY, USA, 2003), ACM, pp. 144–152.

[131] HAN, P., XIE, B., YANG, F., AND SHEN, R. A scalable P2P recommender system based on distributed collaborative filtering. *Expert Systems with Applications 27*, 2 (2004), 203 – 210.

[132] HARCHOL-BALTER, M., AND DOWNEY, A. B. Exploiting process lifetime distributions for dynamic load balancing. *ACM Transactions on Computer Systems (TOCS) 15*, 3 (1997), 253–285.

[133] HARDT, D. The OAuth 2.0 Authorization Framework. `http://www.ietf.org/rfc/rfc6749.txt`, Oct 2012. Last accessed 2014-07-14.

[134] HARDT, D., BUFU, J., AND HOYT, J. OpenID Attribute Exchange 1.0-final. `http://openid.net/specs/openid-attribute-exchange-1_0.html`, Dec 2007. Last accessed 2014-08-17.

[135] HARVEY, N. J. A., JONES, M. B., SAROIU, S., THEIMER, M., AND WOLMAN, A. Skipnet: a scalable overlay network with practical locality properties. In *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems - Volume 4* (Berkeley, CA, USA, 2003), USITS'03, USENIX Association, pp. 9–9.

[136] HENDRIKX, F., AND BUBENDORFER, K. Malleable Access Rights to Establish and Enable Scientific Collaboration. In *eScience (eScience), 2013 IEEE 9th International Conference on* (Beijing, China, October 2013), IEEE, pp. 334–341.

[137] HENDRIKX, F., AND BUBENDORFER, K. Policy Derived Access Rights in the Social Cloud. In *eScience (eScience), 2013 IEEE 9th International Conference on* (Beijing, China, October 2013), IEEE, pp. 365–368.

[138] HENDRIKX, F., BUBENDORFER, K., AND CHARD, R. Reputation systems: A survey and taxonomy. *Journal of Parallel and Distributed Computing* (2014).

[139] HERLOCKER, J. L., KONSTAN, J. A., TERVEEN, L. G., AND RIEDL, J. T. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS) 22*, 1 (2004), 5–53.

[140] HERRERA, O., AND ZNATI, T. Modeling churn in P2P networks. In *Simulation Symposium, 2007. ANSS'07. 40th Annual* (2007), IEEE, pp. 33–40.

[141] HILLEBRAND, C., AND COETZEE, M. Towards Reputation-as-a-Service. In *Information Security for South Africa, 2013* (Aug 2013), pp. 1–8.

[142] HIRSCH, J. E. An Index to Quantify an Individual's Scientific Research Output. *Proceedings of the National Academy of Sciences of the United states of America 102*, 46 (2005), 16569.

[143] HOFFMAN, D. L., NOVAK, T. P., AND PERALTA, M. Building consumer trust online. *Communications of the ACM 42*, 4 (1999), 80–85.

[144] HOFFMAN, K., ZAGE, D., AND NITA-ROTARU, C. A Survey of Attack and Defense Techniques for Reputation Systems. *ACM Computing Surveys 42*, 1 (2009), 1–31.

[145] HORTON, J. J., AND CHILTON, L. B. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM conference on Electronic commerce* (New York, NY, USA, 2010), EC '10, ACM, pp. 209–218.

[146] HOSSFELD, T., LEHRIEDER, F., HOCK, D., OECHSNER, S., DESPOTOVIC, Z., KELLERER, W., AND MICHEL, M. Characterization of BitTorrent swarms and their distribution in the Internet. *Computer Networks 55*, 5 (2011), 1197–1215.

[147] HOUSER, D., AND WOODERS, J. Reputation in auctions: Theory, and evidence from eBay. *Journal of Economics & Management Strategy 15*, 2 (2006), 353–369.

[148] HOWE, J. Crowdsourcing: A definition. `http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html`, June 2006. Last accessed 2014-04-07.

[149] HOWE, J. The rise of crowdsourcing. *Wired magazine 14*, 14 (2006), 1–5.

[150] HU, Y. C., RODNEY, D. A., AND DRUSCHEL, P. Design and scalability of nls, a scalable naming and location service. In *INFOCOM 2002. Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies* (2002), vol. 3, pp. 1218–1227.

[151] HUYNH, T. D., JENNINGS, N. R., AND SHADBOLT, N. Fire: An integrated trust and reputation model for open multi-agent systems. In *ECAI* (2004), vol. 16, p. 18.

[152] HUYNH, T. D., JENNINGS, N. R., AND SHADBOLT, N. R. Developing an integrated trust and reputation model for open multi-agent systems. In *Proceedings of the 7th international workshop on trust in agent societies* (2004), pp. 65–74.

[153] IPEIROTIS, P. Demographics of mechanical turk. *Center for Digital Economy Research, NYU Stern School of Business, Working paper* (2010).

[154] ISMAIL, R., BOYD, C., JØSANG, A., AND RUSSELL, S. An Efficient Off-Line Reputation Scheme Using Articulated Certificates. In *WOSIS-2004: Proceedings of the Second International Workshop on Security in Information Systems* (2004), pp. 53–62.

[155] JERAGH, M., ALQURAISHI, E., AND ALDWAISAN, E. A twitter-based weighted reputation system. *Procedia Computer Science 10*, 0 (2012), 902 – 908. ¡ce:title¿ANT 2012 and MobiWIS 2012¡/ce:title¿.

[156] JIN, X., KRISHNAN, R., AND SANDHU, R. A unified attribute-based access control model covering DAC, MAC and RBAC. In *Data and applications security and privacy XXVI*. Springer, 2012, pp. 41–55.

[157] JOHN, K., BUBENDORFER, K., AND CHARD, K. A Social Cloud for Public eResearch. In *E-Science (e-Science), 2011 IEEE 7th International Conference on* (2011), IEEE, pp. 363–370.

[158] JØSANG, A., AND ISMAIL, R. The Beta Reputation System. In *Proceedings of the 15th bled electronic commerce conference* (Jun 2002), pp. 41–55.

[159] JØSANG, A., ISMAIL, R., AND BOYD, C. A Survey of Trust and Reputation Systems for Online Service Provision. *Decision Support Systems 43*, 2 (2007), 618–644.

[160] JØSANG, A., LUO, X., AND CHEN, X. Continuous ratings in discrete bayesian reputation systems. In *Trust Management II*. Springer, 2008, pp. 151–166.

[161] JOSUTTIS, N. *SOA in Practice*. O'Reilly, 2007.

[162] JR, R. H. S. Electronic document management: Challenges and opportunities for information systems managers. *MIS Quarterly* (1995), 29–49.

[163] JURCA, R., AND FALTINGS, B. An incentive compatible reputation mechanism. *E-Commerce Technology, IEEE International Conference on 0* (2003), 285.

[164] JURCA, R., AND FALTINGS, B. Reputation-based Pricing of P2P Services. In *P2PECON '05: Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems* (New York, NY, USA, 2005), ACM, pp. 144–149.

[165] JURCA, R., AND FALTINGS, B. Minimum payments that reward honest reputation feedback. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce* (New York, NY, USA, 2006), ACM, pp. 190–199.

[166] KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proceedings of the 12th International World Wide Web Conference* (New York, NY, USA, 2003), ACM, pp. 640–651.

[167] KANG, M. H., PARK, J. S., AND FROSCHER, J. N. Access Control Mechanisms for Inter-Organizational Workflow. In *Proceedings of the sixth ACM symposium on Access control models and technologies* (2001), ACM, pp. 66–74.

[168] KANT, K., IYER, R., AND TEWARI, V. A framework for classifying peer-to-peer technologies. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid* (May 2002), p. 368.

[169] KARGER, D., LEHMAN, E., LEIGHTON, T., PANIGRAHY, R., LEVINE, M., AND LEWIN, D. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* (1997), ACM, pp. 654–663.

[170] KARLINS, M., AND ABELSON, H. Persuasion, how opinion and attitudes are changed, 1970.

[171] KARP, A. H., HAURY, H., AND DAVIS, M. H. From ABAC to ZBAC: the evolution of access control models. *Hewlett-Packard Development Company, LP 21* (2009).

[172] KAUFMANN, N., SCHULZE, T., AND VEIT, D. More than fun and money. worker motivation in crowdsourcing - a study on mechanical turk. In *Proceedings of the Seventeenth Americas Conference on Information Systems* (2011).

[173] KAYE, J., HEENEY, C., HAWKINS, N., DE VRIES, J., AND BODDINGTON, P. Data Sharing in Genomics - Re-shaping Scientific Practice. *Nature Reviews Genetics 10*, 5 (2009), 331–335.

[174] KELEHER, P., BHATTACHARJEE, B., AND SILAGHI, B. Are virtualized overlay networks too much of a good thing? In *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds., vol. 2429 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2002, pp. 225–231.

[175] KERSCHBAUM, F., HALLER, J., KARABULUT, Y., AND ROBINSON, P. Pathtrust: A trust-based reputation service for virtual organization formation. In *Trust Management*, K. Stølen, W. Winsborough, F. Martinelli, and F. Massacci, Eds., vol. 3986 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, pp. 193–205.

[176] KINATEDER, M., AND PEARSON, S. A Privacy-Enhanced Peer-to-Peer Reputation System. In *E-Commerce and Web Technologies*, vol. 2738 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2003, pp. 206–215.

[177] KINATEDER, M., AND ROTHERMEL, K. Architecture and Algorithms for a Distributed Reputation System. In *Proceedings of the First International Conference on Trust Management* (2003), Springer-Verlag, pp. 1–16.

[178] KITTUR, A., CHI, E. H., AND SUH, B. Crowdsourcing user studies with mechanical turk. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (2008), ACM, pp. 453–456.

[179] KLEEMANN, F., VOSS, G. G., AND RIEDER, K. Un(der) paid innovators: The commercial utilization of consumer work through crowdsourcing. *Science, Technology & Innovation Studies 4*, 1 (2008), PP–5.

[180] KOBSA, A. Privacy-enhanced web personalization. In *The adaptive web* (2007), Springer-Verlag, pp. 628–670.

[181] KORSGAARD, T. R., AND JENSEN, C. D. Reengineering the wikipedia for reputation. *Electronic Notes in Theoretical Computer Science 244*, 0 (2009), 81 – 94. Proceedings of the 4th International Workshop on Security and Trust Management (STM 2008).

[182] KOUTROULI, E., AND TSALGATIDOU, A. Reputation-based trust systems for P2P applications: design issues and comparison framework. *Trust and Privacy in Digital Business* (2006), 152–161.

[183] KOUTROULI, E., AND TSALGATIDOU, A. Taxonomy of Attacks and Defense Mechanisms in P2P Reputation Systems - Lessons for reputation system designers. *Computer Science Review 6*, 2 (2012), 47–70.

[184] KRUKOW, K., NIELSEN, M., AND SASSONE, V. A framework for concrete reputation-systems with applications to history-based access control. In *Proceedings of the 12th ACM conference on Computer and communications security* (New York, NY, USA, 2005), CCS '05, ACM, pp. 260–269.

[185] KUHN, D. R., COYNE, E. J., AND WEIL, T. R. Adding attributes to role-based access control. *Computer 43*, 6 (2010), 79–81.

[186] KUTZNER, K., AND FUHRMANN, T. Measuring Large Overlay Networks The Overnet Example. In *Kommunikation in Verteilten Systemen (KiVS)*, P. Mller, R. Gotzhein, and J. Schmitt, Eds., Informatik aktuell. Springer Berlin Heidelberg, 2005, pp. 193–204.

[187] LAI, K., FELDMAN, M., STOICA, I., AND CHUANG, J. Incentives for cooperation in peer-to-peer networks, 2003.

[188] LAPLANTE, P. A., ZHANG, J., AND VOAS, J. Distinguishing between Software Oriented Architecture and Software as a Service: What's in a Name? *IEEE IT Professional 10*, 3 (2008), 46–50.

[189] LEISERSON, C. E. Fat-trees: University networks for hardware-efficient supercomputing. *IEEE Transactions on Computers 34* (1985), 892–901.

[190] LESNIEWSKI-LAAS, C. A Sybil-proof one-hop DHT. In *Proceedings of the 1st workshop on Social network systems* (2008), ACM, pp. 19–24.

[191] LIANG, AND CRAMPTON, J. Set covering problems in role-based access control. In *Computer Security–ESORICS 2009*. Springer, 2009, pp. 689–704.

[192] LIAU, C., ZHOU, X., BRESSAN, S., AND TAN, K. Efficient distributed reputation scheme for peer-to-peer systems. *Web and Communication Technologies and Internet-Related Social Issues.HSI 2003* (2003), 172–172.

[193] LIU, L., AND MUNRO, M. Systematic analysis of centralized online reputation systems. *Decision Support Systems 52*, 2 (2012), 438 – 449.

[194] LIU, X., AND XIAO, L. hiREP: Hierarchical Reputation Management for Peer-to-Peer Systems. *International Conference on Parallel Processing 0* (2006), 289–296.

[195] LIU, Z., LIU, Y., AND HE, Y. A two-layered P2P model for semantic service discovery. In *New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on* (May 2010), pp. 41 –46.

[196] LOPEZ, M., VUKOVIC, M., AND LAREDO, J. PeopleCloud service for enterprise crowdsourcing. In *Services Computing (SCC), 2010 IEEE International Conference on* (july 2010), pp. 538 –545.

[197] LORCH, M., PROCTOR, S., LEPRO, R., KAFURA, D., AND SHAH, S. First experiences using XACML for access control in distributed systems. In *Proceedings of the 2003 ACM Workshop on XML Security* (NEw York, NY, USA, 2003), XMLSEC '03, ACM, pp. 25–37.

[198] LUA, K., CROWCROFT, J., PIAS, M., SHARMA, R., AND LIM, S. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys and Tutorials 2* (2005), 72–93.

[199] LV, Q., CAO, P., COHEN, E., LI, K., AND SHENKER, S. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing* (New York, NY, USA, 2002), ICS '02, ACM, pp. 84–95.

[200] MA, H., ZHOU, D., LIU, C., LYU, M. R., AND KING, I. Recommender systems with Social Regularization. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining* (New York, NY, USA, 2011), WSDM '11, ACM, pp. 287–296.

[201] MAHLER, T., AND OLSEN, T. Reputation systems and data protection law. *eAdoption and the Knowledge Economy: Issues, Applications, Case Studies* (2004), 180–187.

[202] MALIK, Z., AND BOUGUETTAYA, A. RATEWeb: Reputation Assessment for Trust Establishment among Web Services. *The VLDB Journal 18*, 4 (2009), 885–911.

[203] MALY, R. J., MISCHKE, J., KURTANSKY, P., AND STILLER, B. Comparison of centralized (client-server) and decentralized (peer-to-peer) networking. Tech. rep., Swiss Federal Institute of Technology, Zurich, Switzerland, 2003.

[204] MANE, S., MOPURU, S., MEHRA, K., AND SRIVASTAVA, J. Network size estimation in a peer-to-peer network, September 2005.

[205] MÁRMOL, F. G., KUHNEN, M. Q., AND PÉREZ, G. M. Enhancing OpenID through a Reputation Framework. In *Autonomic and Trusted Computing*, J. Calero, L. Yang, F. Mármol, L. G. Villalba, A. Li, and Y. Wang, Eds., vol. 6906 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2011, pp. 1–18.

[206] MÁRMOL, F. G., AND PÉREZ, G. M. Towards pre-standardization of trust and reputation models for distributed and heterogeneous systems. *Computer Standards & Interfaces 32*, 4 (2010), 185–196.

[207] MARTI, S., AND GARCIA-MOLINA, H. Taxonomy of Trust: Categorizing P2P Reputation Systems. *Computer Networks 50*, 4 (2006), 472–484.

[208] MARTIN, D., BURSTEIN, M., MCDERMOTT, D., MCILRAITH, S., PAOLUCCI, M., SYCARA, K., MCGUINNESS, D. L., SIRIN, E., AND SRINIVASAN, N. Bringing Semantics to Web Services with OWL-S. *World Wide Web 10*, 3 (2007), 243–277.

[209] MATSUNAGA, A., THOMPSON, A., FIGUEIREDO, R. J., GERMAIN-AUBREY, C. C., COLLINS, M., BEAMAN, R. S., MACFADDEN, B. J., RICCARDI, G., SOLTIS, P. S., PAGE, L. M., AND FORTES, J. A. A Computational and Storage-Cloud for Integration of Biodiversity Collections. In *Proceedings of the 9th IEEE International Conference on eScience (eScience 2013)* (Beijing, China, October 2013), IEEE.

[210] MAXIMILIEN, E. M., AND SINGH, M. P. Toward autonomic web services trust and selection. In *Proceedings of the 2nd international conference on Service oriented computing* (2004), ACM, pp. 212–221.

[211] MAYMOUNKOV, P., AND MAZIÈRES, D. Kademlia: A peer-to-peer information system based on the XOR metric. *Peer-to-Peer Systems* (2002), 53–65.

[212] MCILRAITH, S. A., SON, T. C., AND ZENG, H. Semantic Web Services. *Intelligent Systems, IEEE 16*, 2 (2001), 46–53.

[213] MELNIK, M. I., AND ALM, J. Does a Seller's eCommerce Reputation Matter? Evidence from eBay Auctions. *The journal of industrial economics 50*, 3 (2002), 337–349.

[214] MENASCE, D. A. Composing web services: A QoS view. *Internet Computing, IEEE 8*, 6 (november-december 2004), 88 – 90.

[215] MESTER, L. J. What's the Point of Credit Scoring? *Business review 3* (1997), 3–16.

[216] MICCIANCIO, D. A first glimpse of cryptography's holy grail. *Commun. ACM 53* (March 2010), 96–96.

[217] MIN WANG, C., AND TURNER, D. Extending the wiki paradigm for use in the classroom. In *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on* (2004), vol. 1, IEEE, pp. 255–259.

[218] MOHAISEN, A., TRAN, H., CHANDRA, A., AND KIM, Y. Socialcloud: Using social networks for building distributed computing services. *arXiv preprint arXiv:1112.2254* (2011).

[219] MORI, J., SUGIYAMA, T., AND MATSUO, Y. Real-world oriented information sharing using social networks. In *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work* (2005), ACM, pp. 81–84.

[220] MUI, L., MOHTASHEMI, M., AND HALBERSTADT, A. A Computational Model of Trust and Reputation. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on* (Jan 2002), IEEE, pp. 2431–2439.

[221] MUI, L., MOHTASHEMI, M., AND HALBERSTADT, A. A Computational Model of Trust and Reputation. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on* (Jan 2002), pp. 2431–2439.

[222] MUI, L., MOHTASHEMI, M., AND HALBERSTADT, A. Notions of Reputation in Multi-Agent Systems: A Review. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems* (New York, NY, USA, 2002), ACM, pp. 280–287.

[223] NAH, F. F.-H. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology 23*, 3 (2004), 153–163.

[224] NGUYEN, H. T., ZHAO, W., AND YANG, J. A trust and reputation model based on bayesian network for web services. In *Web Services (ICWS), 2010 IEEE International Conference on* (2010), IEEE, pp. 251–258.

[225] NICKERSON, R., MUNTERMANN, J., VARSHNEY, U., AND ISAAC, H. Taxonomy development in information systems: Developing a taxonomy of mobile applications. In *Proceedings of the European Conference on Information Systems* (2009).

[226] NIELSEN, J. Website Response Times. `http://www.nngroup.com/articles/website-response-times/`, June 2010. Last accessed 2014-10-03.

[227] NIELSON, S., CROSBY, S., AND WALLACH, D. A taxonomy of rational attacks. *Peer-to-Peer Systems IV* (2005), 36–46.

[228] NOVAK, D., AND ZEZULA, P. M-Chord: a Scalable Distributed Similarity Search Structure. In *Proceedings of the 1st International conference on Scalable information systems* (New York, NY, USA, 2006), InfoScale '06, ACM.

[229] NOVOTNY, J., TUECKE, S., AND WELCH, V. An online credential repository for the grid: Myproxy. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing, 2001.* (2001), pp. 104–111.

[230] OBREITER, P., AND NIMIS, J. A taxonomy of incentive patterns. In *Agents and Peer-to-Peer Computing*, G. Moro, C. Sartori, and M. Singh, Eds., vol. 2872 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005, pp. 89–100.

[231] O'BRIEN, L., MERSON, P., AND BASS, L. Quality attributes for service-oriented architectures. In *Proceedings of the international Workshop on Systems Development in SOA Environments* (2007), IEEE Computer Society, p. 3.

[232] ODLYZKO, A. M. Internet growth: Myth and reality, use and abuse. *Journal of Computer Resource Management 102* (2001), 23–27.

[233] O'HARA, K., ALANI, H., KALFOGLOU, Y., AND SHADBOLT, N. Trust strategies for the semantic web. In *Proceedings of the Trust, Security and Reputation Workshop at the ISWC04* (2004), pp. 78–85.

[234] OINN, T., GREENWOOD, M., ADDIS, M., ALPDEMIR, M. N., FERRIS, J., GLOVER, K., GOBLE, C., GODERIS, A., HULL, D., MARVIN, D., ET AL. Taverna: Lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience 18*, 10 (2006), 1067–1100.

[235] OOI, B., LIAU, C., AND TAN, K.-L. Managing Trust in Peer-to-Peer Systems Using Reputation-Based Techniques. In *Advances in Web-Age Information Management*, vol. 2762 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2003, pp. 2–12.

[236] OVASKA, K., LAAKSO, M., HAAPA-PAANANEN, S., LOUHIMO, R., CHEN, P., AITTOMÄKI, V., VALO, E., NÚÑEZ-FONTARNAU, J., RANTANEN, V., KARINEN, S., ET AL. Large-scale data integration framework provides a comprehensive view on glioblastoma multiforme. *Genome Medicine 2*, 9 (2010), 65.

[237] PALLIS, G. Cloud computing: The new frontier of internet computing. *Internet Computing, IEEE 14*, 5 (2010), 70–73.

[238] PALMER, B., BUBENDORFER, K., AND WELCH, I. A protocol for verification of an auction without revealing bid values. *Procedia Computer Science 1*, 1 (2010), 2649–2658.

[239] PANTOLA, A., PANCHO-FESTIN, S., AND SALVADOR, F. TULUNGAN: A Self-Promoting-Resistant Reputation System for Collaborative Web Filtering Systems.

In *The Second International Conference on Cyber Security, Cyber Peacefare and Digital Forensic (CyberSec2013)* (2013), The Society of Digital Information and Wireless Communication, pp. 281–292.

[240] PAPAZOGLOU, M. P., TRAVERSO, P., DUSTDAR, S., AND LEYMANN, F. Service-Oriented Computing: State of the Art and Research Challenges. *Computer 40*, 11 (2007), 38–45.

[241] PARK, J. S., SANDHU, R., AND AHN, G.-J. Role-Based Access Control on the Web. *ACM Transactions on Information and System Security (TISSEC) 4*, 1 (2001), 37–71.

[242] PATEL, J., TEACY, W. L., JENNINGS, N. R., AND LUCK, M. A Probabilistic Trust Model for Handling Inaccurate Reputation Sources. In *Trust Management*. Springer, 2005, pp. 193–209.

[243] PAVLOV, E., ROSENSCHEIN, J. S., AND TOPOL, Z. Supporting privacy in decentralized additive reputation systems. *Trust Management* (2004), 108–119.

[244] PFITZMANN, A., AND KÖHNTOPP, M. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In *Designing privacy enhancing technologies* (2001), pp. 1–9.

[245] PINGEL, F., AND STEINBRECHER, S. Multilateral Secure Cross-Community Reputation Systems for Internet Communities. In *Trust, Privacy and Security in Digital Business*, S. Furnell, S. Katsikas, and A. Lioy, Eds., vol. 5185 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, pp. 69–78.

[246] PITOURA, E., AND SAMARAS, G. Locating objects in mobile computing. *IEEE Transactions on Knowledge and Data Engineering 13* (2001), 571–592.

[247] PLACEK, M., AND BUYYA, R. A Taxonomy of Distributed Storage Systems. Tech. rep., University of Melbourne, Melbourne, Australia, 2006.

[248] PONNEKANTI, S. R., AND FOX, A. SWORD: A Developer Toolkit for Web Service Composition. In *11th World Wide Web Conference (Web Engineering Track)* (2002), pp. 7–11.

[249] POOR, N. Mechanisms of an online public sphere: The website slashdot. *Journal of Computer-Mediated Communication 10*, 2 (2005), 00–00.

[250] POUWELSE, J., GARBACKI, P., EPEMA, D., AND SIPS, H. The bittorrent p2p file-sharing system: Measurements and analysis. *Peer-to-Peer Systems IV* (2005), 205–216.

[251] PRÊTRE, B. Attacks on peer-to-peer networks. Tech. rep., Swiss Federal Institute of Technology, Zurich, Switzerland, 2005.

[252] PRIEBE, T., DOBMEIER, W., AND KAMPRATH, N. Supporting attribute-based access control with ontologies. In *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on* (2006), IEEE, pp. 8–pp.

[253] PUJOL, J. M., SANGÜESA, R., AND DELGADO, J. Extracting reputation in multi agent systems by means of social network topology. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1* (New York, NY, USA, 2002), AAMAS '02, ACM, pp. 467–474.

[254] RAITMAN, R., AUGAR, N., AND ZHOU, W. Employing wikis for online collaboration in the e-learning environment: Case study. In *Information Technology and Applications, 2005. ICITA 2005. Third International Conference on* (2005), vol. 2, IEEE, pp. 142–146.

[255] RAO, J., AND SU, X. A Survey of Automated Web Service Composition Methods. In *Semantic Web Services and Web Process Composition*, J. Cardoso and A. Sheth, Eds., vol. 3387 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 43–54.

[256] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SHENKER, S. A Scalable Content-Addressable Network. In *Proceedings of ACM SIGCOMM 2001* (2001), pp. 161–172.

[257] REBAHI, Y., MUJICA, V., AND SISALEM, D. A reputation-based trust mechanism for ad hoc networks. In *Proceedings of the 10th IEEE Symposium on Computers and Communications* (Washington, DC, USA, 2005), ISCC '05, IEEE Computer Society, pp. 37–42.

[258] RECORDON, D., AND REED, D. OpenID 2.0: a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management* (New York, NY, USA, 2006), DIM '06, ACM, pp. 11–16.

[259] REICH, C., BUBENDORFER, K., BANHOLZER, M., AND BUYYA, R. SLA-Oriented management of containers for hosting stateful web services. In *Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing* (2007).

[260] REN, Y., LI, M., CUI, Y., GUO, C., AND SAKURAI, K. Enhancing Cooperative Behavior for Online Reputation Systems by Group Selection. In *UIC-ATC '09: Proceedings of the 2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing* (Washington, DC, USA, 2009), IEEE Computer Society, pp. 568–573.

[261] Resnick, P., Kuwabara, K., Zeckhauser, R., and Friedman, E. Reputation Systems. *Communications of the ACM 43* (December 2000), 45–48.

[262] Resnick, P., and Zeckhauser, R. Trust among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System. *Advances in Micoeconomics: A Research Annual 11* (2002), 127–157.

[263] Resnick, P., Zeckhauser, R., Swanson, J., and Lockwood, K. The Value of Reputation on eBay: A Controlled Experiment. *Experimental Economics 9* (2003), 79–101.

[264] Rhea, S., Geels, D., Roscoe, T., and Kubiatowicz, J. Handling churn in a DHT. In *Proceedings of the USENIX Annual Technical Conference* (2004), Boston, MA, USA, pp. 127–140.

[265] Ripeanu, M., Foster, I., and Iamnitchi, A. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *arXiv preprint cs/0209028* (2002).

[266] Rosen, J. The Web Means the End of Forgetting. `http://www.nytimes.com/2010/07/25/magazine/25privacy-t2.html?pagewanted=all`, July 2010. Last accessed 2014-04-07.

[267] Rouse, A. C. A preliminary taxonomy of crowdsourcing. In *ACIS 2010 Proceedings* (2010).

[268] Rowstron, A., and Druschel, P. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Middleware* (2001), 329–350.

[269] Rozsenich, C. Challenges of paid crowdsourcing. Presentation given at CrowdNet 2012: 2nd Workshop on Cloud Labor and Human Computation, January 2012.

[270] Ruohomaa, S., Kutvonen, L., and Koutrouli, E. Reputation Management Survey. In *The Second International Conference on Availability, Reliability and Security* (April 2007), ARES 2007, pp. 103–111.

[271] Russello, G., Dong, C., and Dulay, N. A Workflow-based Access Control Framework for e-Health Applications. In *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on* (2008), IEEE, pp. 111–120.

[272] Sabater, J., and Sierra, C. REGRET: Reputation in Gregarious Societies. In *Proceedings of the fifth international conference on Autonomous agents* (New York, NY, USA, 2001), AGENTS '01, ACM, pp. 194–195.

[273] SABATER, J., AND SIERRA, C. Social ReGreT, a Reputation Model based on Social Relations. *SIGecom Exch. 3* (December 2001), 44–56.

[274] SABATER, J., AND SIERRA, C. Review on Computational Trust and Reputation Models. *Artificial Intelligence Review 24* (2005), 33–60.

[275] SAMARATI, P., AND DE VIMERCATI, S. C. Access control: Policies, models, and mechanisms. In *Foundations of Security Analysis and Design*. Springer, 2001, pp. 137–196.

[276] SANDHU, R. S., AND SAMARATI, P. Access control: principle and practice. *Communications Magazine, IEEE 32*, 9 (1994), 40–48.

[277] SARMADY, S. A Survey on Peer-to-Peer and DHT. Tech. rep., School of Computer Science, Universiti Sains Malaysia, 2010.

[278] SAROIU, S., GUMMADI, K. P., DUNN, R. J., GRIBBLE, S. D., AND LEVY, H. M. An analysis of internet content delivery systems. *SIGOPS Oper. Syst. Rev. 36* (December 2002), 315–327.

[279] SCHENK, E., AND GUITTARD, C. Crowdsourcing: What can be outsourced to the crowd, and why ? In *Workshop on Open Source Innovation, Strasbourg, France* (2009).

[280] SCHENK, E., AND GUITTARD, C. Towards a characterization of crowdsourcing practices. *Journal of Innovation Economics*, 1 (2011), 93–107.

[281] SCHLOSSER, A., VOSS, M., AND BRÜCKNER, L. On the simulation of global reputation systems. *Journal of Artificial Societies and Social Simulation 9* (2005), 1.

[282] SCHLOSSER, M., SINTEK, M., DECKER, S., AND NEJDL, W. A scalable and ontology-based P2P infrastructure for semantic web services. In *Proceedings of the Second International Conference on Peer-to-Peer Computing* (2002), P2P 2002, pp. 104 – 111.

[283] SEGAL, D. A Bully Finds a Pulpit on the Web. `http://www.nytimes.com/2010/11/28/business/28borker.html?_r=3&pagewanted=all`, November 2010. Last accessed 2014-04-07.

[284] SELVARAJ, C., AND ANAND, S. A survey on Security Issues of Reputation Management Systems for Peer-to-Peer Networks. *Computer Science Review 6*, 4 (2012), 145 – 160.

[285] SEN, S., AND WANG, J. Analyzing peer-to-peer traffic across large networks. *IEEE/ACM Transactions on Networking (ToN) 12*, 2 (2004), 219–232.

[286] SHEEHAN, K. B. Toward a typology of internet users and online privacy concerns. *The Information Society 18*, 1 (2002), 21–32.

[287] SILBERMAN, M. S., ROSS, J., IRANI, L., AND TOMLINSON, B. Seller's problems in human computation markets. In *Proceedings of the ACM SIGKDD Workshop on Human Computation* (2010), ACM, pp. 18–21.

[288] SIMMHAN, Y. L., PLALE, B., AND GANNON, D. Karma2: Provenance management for data-driven workflows. *International Journal of Web Services Research (IJWSR) 5*, 2 (2008), 1–22.

[289] SINGH, A., AND LIU, L. TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems. In *Proceedings of the third international conference on Peer-to-Peer Computing* (2003), P2P 2003, pp. 142–149.

[290] SINGH, A., NGAN, T., DRUSCHEL, P., AND WALLACH, D. S. Eclipse attacks on overlay networks: Threats and defenses. In *IEEE INFOCOM* (2006).

[291] SINGHAL, A. Being Bad to your Customers is Bad for Business. `http://googleblog.blogspot.com/2010/12/being-bad-to-your-customers-is-bad-for.html`, December 2010. Last accessed 2014-04-07.

[292] SONG, S., HWANG, K., ZHOU, R., AND KWOK, Y.-K. Trusted P2P Transactions with Fuzzy Reputation Aggregation. *Internet Computing, IEEE 9*, 6 (2005), 24–34.

[293] SRIVATSA, M., XIONG, L., AND LUI, L. TrustGuard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks. In *Proceedings of the 14th international conference on World Wide Web* (New York, NY, USA, 2005), WWW '05, ACM, pp. 422–431.

[294] STEINBRECHER, S. Design options for privacy-respecting reputation systems within centralised internet communities. In *Security and Privacy in Dynamic Environments*, S. Fischer-Hübner, K. Rannenberg, L. Yngström, and S. Lindskog, Eds., vol. 201 of *IFIP International Federation for Information Processing*. Springer Boston, 2006, pp. 123–134.

[295] STEINBRECHER, S. Privacy-respecting reputation system for future internet communities. In *Proceedings of the European e-Identity Conference on Managing Employee, Citizen & Private Identities* (2008), Citeseer.

[296] STEINBRECHER, S. The need for interoperable reputation systems. In *Open Research Problems in Network Security*, J. Camenisch, V. Kisimov, and M. Dubovitskaya, Eds.,

vol. 6555 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2011, pp. 159–169.

[297] STEINBRECHER, S., GRO, S., AND MEICHAU, M. Jason: A scalable reputation system for the semantic web. In *Emerging Challenges for Security, Privacy and Trust*, D. Gritzalis and J. Lopez, Eds., vol. 297 of *IFIP Advances in Information and Communication Technology*. Springer Boston, 2009, pp. 421–431.

[298] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A Scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review 31*, 4 (2001), 149–160.

[299] SUE, D. W. Multidimensional facets of cultural competence. *The Counseling Psychologist 29*, 6 (2001), 790–821.

[300] SUN, P.-L., AND KU, C.-Y. Review of threats on trust and reputation models. *Industrial Management & Data Systems 114*, 3 (2014), 8–8.

[301] TAVAKOLIFARD, M., KNAPSKOG, S. J., AND HERRMANN, P. Trust Transferability among Similar Contexts. In *Q2SWinet '08: Proceedings of the 4th ACM symposium on QoS and security for wireless and mobile networks* (New York, NY, USA, 2008), ACM, pp. 91–97.

[302] TENOPIR, C., ALLARD, S., DOUGLASS, K. L., AYDINOGLU, A. U., WU, L., READ, E., MANOFF, M., AND FRAME, M. Data Sharing by Scientists: Practices and Perceptions. *PLoS One 6*, 6 (2011).

[303] THAUFEEG, A. M., BUBENDORFER, K., AND CHARD, K. Collaborative eResearch in a Social Cloud. In *E-Science (e-Science), 2011 IEEE 7th International Conference on* (2011), IEEE, pp. 224–231.

[304] TIAN, C., AND YANG, B. R^{2}trust, a reputation and risk based trust management framework for large-scale, fully decentralized overlay networks. *Future Generation Computer Systems 27*, 8 (2011), 1135 – 1141.

[305] TONG, S. T., VAN DER HEIDE, B., LANGWELL, L., AND WALTHER, J. B. Too Much of a Good Thing? The Relationship Between Number of Friends and Interpersonal Impressions on Facebook. *Journal of Computer-Mediated Communication 13*, 3 (2008), 531–549.

[306] TONG, X., AND ZHANG, W. Group Trust and Group Reputation. In *ICNC '09: Proceedings of the 2009 Fifth International Conference on Natural Computation* (Washington, DC, USA, 2009), IEEE Computer Society, pp. 561–565.

[307] TORMO, G. D., MÁRMOL, F. G., AND PÉREZ, G. M. Towards the integration of reputation management in OpenID. *Computer Standards & Interfaces 36*, 3 (2014), 438 – 453.

[308] TURNER, M., BUDGEN, D., AND BRERETON, P. Turning software into a service. *Computer 36*, 10 (2003), 38–44.

[309] URDANETA, G., PIERRE, G., AND STEEN, M. V. A survey of DHT security techniques. *ACM Computing Surveys (CSUR) 43*, 2 (2011), 8.

[310] VAIDYA, J., ATLURI, V., AND GUO, Q. The role mining problem: finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM symposium on Access control models and technologies* (2007), ACM, pp. 175–184.

[311] VAN STEEN, M., HAUCK, F. J., BALLINTIJN, G., AND TANENBAUM, A. S. Algorithmic design of the globe wide-area location service. *The Computer Journal 41*, 5 (1998), 297.

[312] VAN STEEN, M., HOMBURG, P., AND TANENBAUM, A. S. The architectural design of Globe: A wide-area distributed system. Tech. rep., Technical Report IR-422, Department of Mathematics and Computer Science, Vrije Universiteit, 1997.

[313] VAN STEEN, M., HOMBURG, P., AND TANENBAUM, A. S. Globe: a wide area distributed system. *IEEE Concurrency 7*, 1 (1999), 70–78.

[314] VOLLBRECHT, J., CALHOUN, P., FARRELL, S., GOMMANS, L., GROSS, G., DE BRUIJN, B., DE LAAT, C., HOLDREGE, M., AND SPENCE, D. AAA Authorization Framework. `http://www.ietf.org/rfc/rfc2904.txt`, August 2000. Last accessed 2014-07-14.

[315] VON DER TRENCK, F. F. *The life of baron Frederick Trenck, containing his adventures, his cruel and excessive sufferings during ten years' imprisonment at the fortress of Magdeburg, by command of the late king of Prussia.* Timothy Bedlington, No. 31, Washington Street, Boston, 1828.

[316] VOSS, M., HEINEMANN, A., AND MUHLHAUSER, M. A privacy preserving reputation system for mobile information dissemination networks. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks* (2005), SECURECOMM '05, IEEE Computer Society, pp. 171–181.

[317] VU, L.-H., HAUSWIRTH, M., AND ABERER, K. QoS-Based Service Selection and Ranking with Trust and Reputation Management. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, R. Meersman and Z. Tari, Eds.,

vol. 3760 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005, pp. 466–483.

[318] VUKOVIC, M. Crowdsourcing for enterprises. In *2009 World Conference on Services* (july 2009), pp. 686 –692.

[319] WALKER, I. Reputation scraper - social media. Master's thesis, Victoria University of Wellington, 2013.

[320] WALTHER, J. B., VAN DER HEIDE, B., KIM, S.-Y., WESTERMAN, D., AND TONG, S. T. The Role of Friends' Appearance and Behavior on Evaluations of Individuals on Facebook: Are We Known by the Company We Keep? *Human Communication Research 34*, 1 (2008), 28–49.

[321] WANG, C., AND LI, B. Peer-to-Peer Overlay Networks: A Survey. Tech. rep., The Hong Kong University of Science and Technology, 2003.

[322] WANG, L., AND KANGASHARJU, J. Measuring large-scale distributed systems: case of bittorrent mainline dht. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on* (2013), IEEE, pp. 1–10.

[323] WANG, L., WIJESEKERA, D., AND JAJODIA, S. A logic-based framework for attribute based access control. In *Proceedings of the 2004 ACM workshop on Formal methods in security engineering* (2004), ACM, pp. 45–55.

[324] WANG, Y., AND NAKAO, A. Poisonedwater: An improved approach for accurate reputation ranking in P2P networks. *Future Generation Computer Systems 26*, 8 (2010), 1317 – 1326.

[325] WANG, Y., AND VASSILEVA, J. Trust and reputation model in peer-to-peer networks. In *Proceedings of the 2003 third International Conference on Peer-to-Peer Computing (P2P 2003).* (2003), IEEE, pp. 150–157.

[326] WANG, Y., AND VASSILEVA, J. A review on trust and reputation for web service selection. In *Distributed Computing Systems Workshops, 2007. ICDCSW'07. 27th International Conference on* (2007), IEEE, pp. 25–25.

[327] WANG, Y., AND VASSILEVA, J. Toward Trust and Reputation Based Web Service Selection: A Survey. *International Transactions on Systems Science and Applications 3*, 2 (2007), 118–132.

[328] WILLIAMS, C., HUIBONHOA, P., HOLLIDAY, J., HOSPODOR, A., AND SCHWARZ, T. Redundancy management for P2P storage. In *Seventh IEEE International Symposium on Cluster Computing and the Grid* (May 2007), CCGRID 2007, pp. 15 – 22.

[329] WINDLEY, P. J., DALEY, D., CUTLER, B., AND TEW, K. Using reputation to augment explicit authorization. In *Proceedings of the 2007 ACM workshop on Digital identity management* (New York, NY, USA, 2007), DIM '07, ACM, pp. 72–81.

[330] WINDLEY, P. J., TEW, K., AND DALEY, D. A framework for building reputation systems. *WWW 2007* (2007), 8–12.

[331] WONGRUJIRA, K., AND SENEVIRATNE, A. Monetary incentive with reputation for virtual market-place based P2P. In *CoNEXT '05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology* (New York, NY, USA, 2005), ACM, pp. 135–145.

[332] WU, S., SHETH, A., MILLER, J., AND LUO, Z. Authorization and access control of application data in workflow systems. *Journal of Intelligent Information Systems 18*, 1 (2002), 71–94.

[333] WU, X., HE, J., AND XU, F. A Group-Based Reputation Mechanism for Mobile P2P Networks. In *GPC '09: Proceedings of the 4th International Conference on Advances in Grid and Pervasive Computing* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 410–421.

[334] WU, Y., YAN, C., DING, Z., LIU, G., WANG, P., JIANG, C., AND ZHOU, M. A novel method for calculating service reputation. *Automation Science and Engineering, IEEE Transactions on 10*, 3 (July 2013), 634–642.

[335] XIONG, L., AND LIU, L. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. *IEEE Transactions on Knowledge and Data Engineering 16*, 7 (2004), 843–857.

[336] YAGUE, M., MANA, A., LOPEZ, J., AND TROYA, J. Applying the semantic web layers to access control. In *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on* (2003), pp. 622–626.

[337] YAGÜE, M. I., MANA, A., AND LOPEZ, J. A metadata-based access control model for web services. *Internet Research 15*, 1 (2005), 99–116.

[338] YAN, Z., CHEN, Y., AND SHEN, Y. A practical reputation system for pervasive social chatting. *Journal of Computer and System Sciences 79*, 5 (2013), 556–572.

[339] YAN, Z., CHEN, Y., AND SHEN, Y. PerContRep: a Practical Reputation System for Pervasive Content Services. *The Journal of Supercomputing* (2014), 1–24.

[340] YANG, B., ZHENG-DING, L., BAO-HUA, H., RUI-XUAN, L., HE-PING, H., AND SONG-FENG, L. Quorum-based optimistic concurrency control in replicated DHTs.

In *Information Engineering and Electronic Commerce, 2009. IEEC'09. International Symposium on* (2009), IEEE, pp. 40–45.

[341] YOU, W., LIU, L., XIA, M., AND LV, C. Reputation inflation detection in a Chinese C2C market. *Electronic Commerce Research and Applications 10*, 5 (2011), 510 – 519.

[342] YU, A. P., AND VUONG, S. T. MOPAR: a Mobile Peer-to-Peer Overlay Architecture for Interest Management of Massively Multiplayer Online Games. In *Proceedings of the international workshop on Network and operating systems support for digital audio and video* (New York, NY, USA, 2005), NOSSDAV '05, ACM, pp. 99–104.

[343] YU, B., AND SINGH, M. P. A social mechanism of reputation management in electronic communities. In *Proceedings of Fourth International Workshop on Cooperative Information Agents* (2000), pp. 154–165.

[344] YU, B., AND SINGH, M. P. Incentive mechanisms for peer-to-peer systems. In *Proceedings of the Second International Workshop on Agents and Peer-to-Peer Computing* (2003), pp. 77–88.

[345] YU, H., GIBBONS, P. B., KAMINSKY, M., AND XIAO, F. SybilLimit: A Near-Optimal Social Network Defense against Sybil Attacks. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on* (2008), IEEE, pp. 3–17.

[346] YU, S., WANG, C., REN, K., AND LOU, W. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM, 2010 Proceedings IEEE* (2010), IEEE, pp. 1–9.

[347] YU, T., ZHANG, Y., AND LIN, K.-J. Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints. *ACM Trans. Web 1* (May 2007).

[348] YUAN, E., AND TONG, J. Attributed based access control (ABAC) for web services. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on* (2005), IEEE.

[349] YUAN, Y., RUOHOMAA, S., AND XU, F. Addressing common vulnerabilities of reputation systems for electronic commerce. *Journal of theoretical and applied electronic commerce research 7*, 1 (2012), 1–20.

[350] YURCIK, W. J., BONILLA, R. F., STAGELL, A., AND BASNEY, J. Credential wallets: A classification of credential repositories highlighting myproxy.

[351] ZACHARIA, G., MOUKAS, A., AND MAES, P. Collaborative reputation mechanisms for electronic marketplaces. *Decision Support Systems 29*, 4 (2000), 371 – 388.

[352] ZENG, L., BENATALLAH, B., DUMAS, M., KALAGNANAM, J., AND SHENG, Q. Z. Quality Driven Web Services Composition. In *Proceedings of the 12th international conference on World Wide Web* (2003), ACM, pp. 411–421.

[353] ZENG, L., BENATALLAH, B., NGU, A. H. H., DUMAS, M., KALAGNANAM, J., AND CHANG, H. QoS-Aware middleware for web services composition. *IEEE Transactions on Software Engineering 30*, 5 (2004), 311–327.

[354] ZENG, L., BENATALLAH, B., NGUYEN, P., AND NGU, A. H. AgFlow: Agent-based Cross-Enterprise Workflow Management System. In *Proceedings of the 27th International Conference on Very Large Data Bases* (San Francisco, CA, USA, 2001), VLDB '01, Morgan Kaufmann Publishers Inc., pp. 697–698.

[355] ZHANG, H., WU, W., AND LI, Z. Open Social based Group Access Control Framework for e-Science Data Infrastructure. In *E-Science (e-Science), 2012 IEEE 8th International Conference on* (2012), pp. 1–8.

[356] ZHANG, Z., SHI, S.-M., AND ZHU, J. SOMO: Self-Organized Metadata Overlay for Resource Management in P2P DHT. In *Peer-to-Peer Systems II*, vol. 2735 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2003, pp. 170–182.

[357] ZHAO, B. Y., HUANG, L., STRIBLING, J., RHEA, S. C., JOSEPH, A. D., AND KUBIATOWICZ, J. D. Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications 22* (2004), 41–53.

[358] ZHAO, H., AND LI, X. H-Trust: A Group Trust Management System for Peer-to-Peer Desktop Grid. *Journal of Computer Science and Technology 24* (2009), 833–843.

[359] ZHAO, H., YANG, X., AND LI, X. An incentive mechanism to reinforce truthful reports in reputation systems. *Journal of Network and Computer Applications 35*, 3 (2012), 951 – 961. ¡ce:title¿Special Issue on Trusted Computing and Communications¡/ce:title¿.

[360] ZHAO, Y., AND ZHU, Q. Evaluation on crowdsourcing research: Current status and future direction. *Information Systems Frontiers* (2012), 1–18.

[361] ZHENG, Z., AND LYU, M. R. WS-DREAM: A Distributed Reliability Assessment Mechanism for Web Services. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on* (2008), IEEE, pp. 392–397.

[362] ZHENG, Z., AND LYU, M. R. A QoS-Aware Fault Tolerant Middleware for Dependable Service Composition. In *Dependable Systems Networks, 2009. DSN'09. IEEE/IFIP International Conference on* (2009), IEEE, pp. 239–248.

[363] ZHENG, Z., MA, H., LYU, M. R., AND KING, I. WSrec: A collaborative Filtering Based Web Service Recommender System. In *Web Services, 2009. ICWS 2009. IEEE International Conference on* (2009), IEEE, pp. 437–444.

[364] ZHOU, R., AND HWANG, K. PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. *IEEE Transactions on Parallel and Distributed Systems 18* (2007), 460–473.

[365] ZITTRAIN, J. Ubiquitous human computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 366*, 1881 (2008), 3813–3821.

[366] ZUCKERBERG, M. Facebook across the Web. `https://www.facebook.com/notes/facebook/facebook-across-the-web/41735647130`, December 2008. Last accessed 2014-04-07.

Glossary

Churn The rate at which peers enter and leave a network.

Co-authorship The state of two (or more) authors having worked together on a single manuscript. For example, if Alice and Bob write a manuscript together, they are considered co-authors.

Context The domain in which information was generated. Most reputation systems employ a single, or personal, context.

Contextual attributes Data that provides additional information about the context in which reputation was generated.

DBLP The DBLP Computer Science Bibliography. An online bibliography of computer science journals and conferences. DBLP makes available its database of authors and papers, allowing the derivation of co-authorship information.

DHT Distributed Hash Table. This concept is further discussed in Section A.3.

Entity An entity describes an independent agent. Both people and machines (or services) are considered to be entities in this thesis.

GP A GRAFT Provider. This means an OpenID Provider (OP) combined with GRAFT profile storage.

GRAFT Generalised Recommendation Framework. The framework that is developed and evaluated in this thesis.

Kademlia An implementation of a DHT, discussed further in Section A.3.1.

Malicious peers Peers that try to subvert the network for their own gain.

MAS Microsoft Academic Search.

OpenID A decentralised authentication protocol that allows an entity to maintain a single digital identity. OpenID is further discussed in Section A.1.

OpenRep GRAFT OpenID extensions that support discovery and exchange of recommendation profiles. Please see Section A.2 for more information.

OP An OpenID Provider. An OP authenticates the entities that are registered with it. Please see Section A.1 for more information.

Peer A single participant in a network of peers. Each peer is equal to every other peer and may act as both a client and a server to other peers.

P2P Peer-to-Peer. A distributed network of peers where resources are coordinated amongst peers, without any centralised control.

PostgreSQL An opensource database compliant with SQL standards.

Recommendation Recommendations are endorsements of the target agent by a third party. These endorsements are based on the target's characteristics, capabilities, affiliations, and reputation.

Reputation According to the Collins English Dictionary, reputation is *"the estimation in which a person or thing is generally held; opinion"*. Reputation is discussed in depth in Chapter 2.

RP A Relying-Party provides services to entities. An RP will authenticate entities with their OPs (or GPs) using the OpenID protocol. Please see Section A.1 for more information.

XRDS XML-based document that allows for the discovery of web-based resources. Utilised by the OpenID protocol to enumerate authentication service locations. Please see Section A.1 for more information.

This thesis was written using a 1989 IBM Model M keyboard.