Res. Lett. Inf. Math. Sci., 2010, Vol. **14**, pp. 1–16
Available online at http://iims.massey.ac.nz/research/letters/

Small-World Networks, Distributed Hash Tables and the e-Resource Discovery Problem

A. Leist, K. A. Hawick
Computer Science,
Institute of Information and Mathematical Sciences,
Massey University
Albany, Auckland, New Zealand
Tel.:+64 9 414-0800
Fax: +64 9 441-8181
Email: {a.leist, k.a.hawick}@massey.ac.nz

Resource discovery is one of the most important underpinning problems behind producing a scalable, robust and efficient global infrastructure for e-Science. A number of approaches to the resource discovery and management problem have been made in various computational grid environments and prototypes over the last decade. Computational resources and services in modern grid and cloud environments can be modelled as an overlay network superposed on the physical network structure of the Internet and World Wide Web. We discuss some of the main approaches to resource discovery in the context of the general properties of such an overlay network. We present some performance data and predicted properties based on algorithmic approaches such as distributed hash table resource discovery and management. We describe a prototype system and use its model to explore some of the known key graph aspects of the global resource overlay network - including small-world and scale-free properties.

Keywords: Semantic Metadata; Small-World; Network; DHT; Routing; P2P; Semantic Web; Cloud Computing

1 Introduction

The general notion of e-Science is the use of connected computational data, services and resources to enable and accelerate the mission of science in all sorts of disciplinary areas beyond those associated with development of the computational infrastructure. As e-Science ideas are increasingly used and up-taken the support infrastructure for scientists to collect data, share data, process and analyse data has to be scalable, robust and efficient and in general, "successful e-Science" requires a global support infrastructure comprising both appropriate hardware but also distributed computing software.

The e-Science infrastructure problem has grown out of networked super-computing and parallel computing [1], broadband networking [2], meta-computing [3] and latterly grid [4] and cloud [5] computing research communities. It appears that e-Science has been embraced by the whole scientific community in a very natural way as global computing infrastructure has become widely available, and easily used by all scientists.

Apart from the general software and systems complexity issues behind setting up global e-Science prototypes, one of the most challenging specific problems has been to find a way for users, software agents and components of the e-Science endeavour to discover resources. Resources might be computational services, data, analysis sub-systems, reduced data, provenance systems, bulk storage engines, supercomputers or indeed other human users.

There continues to be particular interest in small-world [6] and related community network problems [7] that arise from socio-technical applications. In the context of e-Science support the resource discovery issue is at the vanguard of these application problems in that it represents a global-scale application of widespread interest and it also involves computer scientists as both users and as those most likely to be able to contribute theoretically and practically to solutions.

Some typical e-Science scenarios involve arbitrary searches for resources, or more typically searches for details within a partially known sub-community of users such as: the astronomical community [8]; or the particle physics community; or indeed some other socio-technical well-defined sub-group of users and service suppliers. Resource matching [9, 10] is required at a number of levels since a search might initially pick up some suitable short-list of resources that might require subsequent refinement by secondary criteria such as cost, availability, speed of response and so forth.

In addition to the core resource discovery problem, there are also specific problems associated with naming resources, but good frameworks such as RDF [11] and XML-based [12, 13] systems for this have emerged from research work on prototypes. The resource discovery problem itself is still not completely solved although some successful ideas and approaches have emerged in recent years.

In section 2 we review some of the main approaches to resource discovery and its critical role in e-Science support infrastructure. As discussed above, resource discovery needs to be supported by some sort of distributed meta-data infrastructure. In section 3 we discuss the main overlay network ideas for a distributed meta-data storage system including scalability, efficiency and robustness. We describe our prototype implementation and ideas for a distributed meta-data storage system in section 4 including the use and role of RDF, domain management and routing table management strategies. In section 5 we present some network analysis results covering routing table configurations, domain sizes, scaling and clustering. We offer a discussion of our results and findings in the context of present and future resource discovery for e-Science support infrastructure in section 6 and some general conclusions in section 7.

2 Approaches to Resource Discovery

Resource discovery can be modelled as a graph or network problem, whereby a partially connected graph of resources can be modelled as a set of overlapping overlay networks [14] superposed on the network infrastructure of the global Internet and World Wide Web. The problem then becomes how to match resources to user and agent requests [15]. This problem has some interesting implications for the way socio-technical scientific communities form and interact globally. Abstracting the general resource discovery notion into a more general graph problem allows us to study the problem in its generality and think in terms of global patterns and trends without needing to know the details of particular infrastructure grids or the usage needs of particular communities.

Approaches to storing, distributing and matching against resource description information and resource requests has been tackled in a number of interesting ways. In the early stages of e-Science in the 1990's, simple approaches such as flat data bases, simple gossip protocols [16], search agents and other ideas borrowed from Internet Domain Name Systems (DNS) [17] searching have been used. Existing software technologies such as the lightweight directory access protocol (LDAP) have also been employed [18]. A number of prototype grid and e-Science infrastructure projects [19, 20, 21] have provided an experimental basis for different resource discovery algorithms and some simulations have also thrown light on the problem [22, 23]

Over the last decade it has become clear that simpler algorithms and ideas are no longer adequate and do not scale well nor support the very dynamic changes required for resource discovery on a robust e-Science support infrastructure [24]. A number of investigations have been made using agent-based approaches [25, 26] and these have opened up the possibilities for loosely coupled wide area distributed discovery systems. Decentralised resource discovery approaches [27, 28] involving sophisticated distributed hash tables (DHT) and communicating peer-to-peer [29] and hierarchical peer-based systems [30] are now being widely adopted [31]. There are a number of algorithmic variations possible within such a framework, including various forms of cache duplication and distributed management [32] to optimise speed and performance. We discuss DHT ideas more fully in the remainder of this paper.

3 Network Properties

The underlying network structure of a distributed metadata storage system has to fulfil certain requirements. These are **scalability** to large amounts of data and messages querying, storing or updating this information; **efficiency** when it comes to the delivery of messages and the maintenance of the network infrastructure; as well as **robustness** against churn—"the continuous process of node arrival and departure" [33]—and deliberate attacks.

The properties of complex networks are being studied in many scientific fields. Biologists are interested in metabolic networks [34, 35, 36]; sociologists investigate scientific collaboration networks [37, 38], the structure of criminal organisations [39] or many other types of social networks [6, 40, 41]; and computer scientists are interested in computer networks like the World-Wide Web [42] or in our case overlay networks for the storage of semantic metadata. As diverse as these networks may appear, many of them share certain properties that can be used to categorise them. We are especially interested in networks that show the characteristics of small-world or scale-free graphs.

Graphs are commonly used to represent networks. A graph G consists of a set of vertices V (the nodes of the network) and a set of either edges E (undirected connections between vertices) or arcs A (directed connections between vertices). Graphs with a power-law degree distribution, for which the probability of a node having k links follows $P(k) \sim k^{-\gamma}$, are called scale-free. The exponent γ typically lies between 2.5 and 3.0 [43, 44]. Networks that show the small-world effect have the property that the mean shortest paths between any two nodes (i.e. the mean geodesic distance) scales logarithmically or slower with the network size for fixed degree k [45] (see section 5.3). They are highly clustered (see section 5.4), like a regular graph, but retain the small mean geodesic distance characteristic of random graphs [46].

3.1 Scalability

A network that is supposed to support large amounts of data and varying numbers of concurrent users, possibly in the millions, has to scale well with the current requirements. Distributed peer-to-peer based systems are more flexible than centralised networks and much cheaper to maintain. One such system is the open source Distributed Hash Table (DHT) Bamboo [47], which we use in modified form for our overlay network. The Bamboo DHT does not require any centralised services. Any node in the system acts as a gateway for new nodes. The available resources grow with the system size, since every participant is not only a consumer but also a provider of storage space and can act as a router for messages that are not addressed to itself.

Section 5.3 shows how our system scales with increasing network size in regards to the actual number of hops needed to deliver messages as well as the number of hops that would be needed if the whole network structure was known to every node (i.e. the theoretical minimum for the given network structure).

3.2 Efficiency

In a peer-to-peer system, messages can usually not be sent directly from the source node to the destination node, because there is no global registry that can be queried for the IP address of a certain node in the namespace of the overlay network. The message has to be routed towards the destination based on the local knowledge of the node currently processing it. To maximise performance and reduce network usage, the number of hops and the latencies of the nodes in the chosen route have to be minimised.

The maintenance of the network infrastructure is mainly done by the original implementation of the Bamboo DHT. This includes the protocols used to join the overlay network, keep the basic ring substrate maintained and manage the stored data. Bamboo maintains a ring substrate to ensure that every message can be routed towards its destination even if no suitable entry can be found in the routing table. Every node keeps track of its nearest neighbours in the namespace of the DHT up to a distance that can be specified in the configuration. This set of neighbours is called the leaf set of a node. The goal is to minimise the communication overhead needed for these tasks while still keeping the system's infrastructure up and the stored data accessible.

Section 4 explains how the routing is done in our system and section 5 analyses how it performs in a simulated environment.

3.3 Robustness

Decentralised peer-to-peer networks have no single point of failure, which is an advantage over centralised networks. However, that does not mean that they are necessarily more robust against churn or deliberate attacks. Due to the lack of a global registry, the task of keeping track of available resources is more complex than in a centralised system and more sophisticated mechanisms are necessary to route messages between nodes. Especially under high churn, with many nodes only participating for a short amount of time, quick failure recovery from broken routes and nodes that are no longer available is important.

Decentralised networks can break apart into disjoint clusters when the number of random failures is high or if they are the subject of targeted attacks. The network structure plays an important role in the vulnerability of the network. Previous research [48, 34, 49, 50, 39] has shown that scale-free networks that consist of many nodes with low degrees and very few highly connected hub nodes are robust against random failures, because the likelihood that a hub node fails by random chance is low. However, attacks targeting the hub nodes can be devastating to such a network, as the network diameter rises sharply until the network fragments into isolated clusters if enough hub nodes fail. Both scale-free networks and small-world networks have also been shown to be vulnerable to bridge attacks. Bridges are nodes with a high betweenness centrality, that is, nodes through which pass many shortest paths.

4 The Prototype System

Our implementation of a distributed metadata storage system [51] is based on Atlas [20], which itself is based on the Bamboo DHT. Atlas can store metadata from Resource Description Framework (RDF) [11] and RDF Schema (RDFS) [12] documents in the DHT. The RDF Query Language (RQL) is supported to describe and execute queries on the stored metadata.

We have replaced the Bamboo routing table, which maintains the adjacency-list (i.e. the neighbours) of a node, with an implementation that is based on the concepts of small-world network models.

4.1 RDF Metadata in the DHT

Atlas uses the Query Chain (QC) algorithm [21] to store RDF triples in the DHT. Each triple is stored on three nodes using three keys, the hash values generated from the subject, the predicate and the object of the triple. This makes it possible to find matching triples when only one or two parts of it are specified in the query.

4.1.1 Domains

Our prototype introduces the concept of domains and extends the QC algorithm accordingly. Every node in the network is a member of one domain, which is specified as a parameter when the node is started. The domains represent the clusters that are characteristic for small-world environments [45, 52]. The idea is to group nodes that have a semantic relationship, for example web services that offer computing or storage resources to interested customers, as well as the nodes of those customers that are mainly interested in these types of services, into the same domain. As sections 5.2 and 5.3 will show, the performance of the network can be increased substantially when messages are sent within the same domain. The domains make it possible to organise the nodes in the network into groups.

The goal is to store related triples, which are likely to be part of the same query, in the same domain. This reduces the distance between these triples in the identifier namespace of the DHT and allows queries to be executed faster due to a decreased number of hops needed to route the messages to the relevant nodes in the domain.

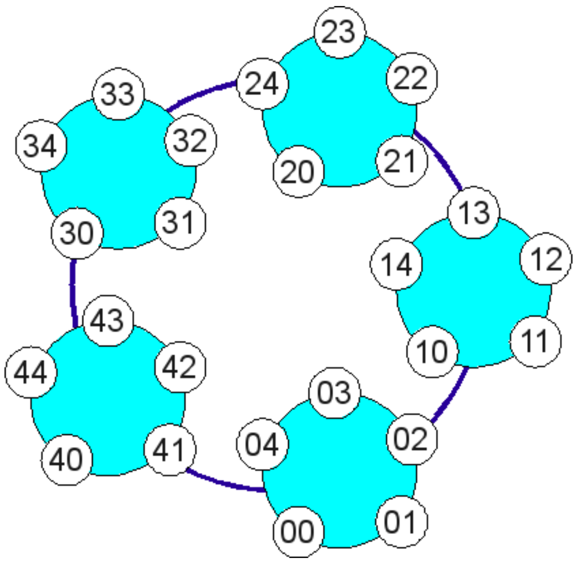

Figure 1: The prototype system introduces domains to the DHT. Every node has a domain (or global) identifier as well as a local identifier. In this example, the first digit of the node ID identifies the domain and the second one identifies the local node. In the actual implementation, the identifiers are each 160 bits long. Hence, the total namespace of the DHT is 320 bits.

When a new RDF document is to be stored, then the system first determines the types (a resource can be an instance of multiple classes in RDF) of each subject. Non-typed resources are discouraged, but we will describe how the system handles such triples later on. For now we assume that the resource is typed and select any one of the found classes. The current implementation expects that the class is identified by an http-address. The protocol and domain part of the URI are used as the domain identifier. This means that the system currently assumes that related concepts are defined under the same domain name. The domain identifier `http://example.com` would be extracted from the class URI `http://example.com/terms/Person`. Unrelated concepts that are defined under the same second-level domain can be distinguished by using sub-domains or by modifying the generator of the domain identifiers to include parts of the directory path.

In the next step, the domain identifiers of all other types of the subject, the subjects themselves and the predicates are extracted as well. A reference triple has to be generated for each of these domain identifiers that is distinct from the one extracted from the chosen type URI. They are needed in order to be able to evaluate query path expressions that do not specify the type of the subject or that only specify types that were not used to extract the main domain identifier. The reference triples declare that the respective resource R is used in domain D of the previously selected main type. The reference triples have the following form:

```
R <http://example.com/terms/usedInDomain> D
```

These reference triples are then stored in the domains identified by the domain identifiers which were extracted from the respective resources.

When triples with non-typed subjects have to be stored, then they are grouped by subject and the domain identifiers of the respective predicate URIs are extracted. If one domain identifier is extracted more often then others, then that one is selected to identify the main domain for this group of triples. Otherwise, it has to be selected randomly. Reference triples for distinct domain identifiers extracted from subjects and predicates are generated as described before. It may appear strange to choose the predicate over the subject URI when the subject is not typed. The reason is that a subject can be a blank node in the RDF graph [53], and blank nodes do not have an URI.

4.1.2 Storing and Querying RDF Metadata

This section describes how the following sample RDF document is stored in the DHT and how the information can later on be queried.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf= "http://www.w3.org/.../22-rdf-syntax-ns#"
  xmlns:contact= "http://www.w3.org/.../contact#">

   <contact:Person
     rdf:about="http://example.com/contact#me">
      <contact:firstName>Arno</contact:firstName>
      <contact:lastName>Leist</contact:lastName>
      <contact:mailbox  rdf:resource="mailto:a.leist@massey.ac.nz"/>
   </contact:Person>
</rdf:RDF>
```

This RDF/XML document can be passed to the storage system, which extracts the RDF triples from it. It determines the types of the subject (`http://example.com/contact#me`), in this case the single superclass is identified by the URI `http://www.w3.org/.../contact#Person`. This URI is used to extract the target domain `http://www.w3.org`. All extracted triples are stored in this domain:

```
<http://example.com/contact#me>
   <http://www.w3.org/.../contact#firstName>
      "Arno" .

<http://example.com/contact#me>
   <http://www.w3.org/.../contact#lastName>
      "Leist" .

<http://example.com/contact#me>
   <http://www.w3.org/.../contact#mailbox>
      <mailto:a.leist@massey.ac.nz> .

<http://example.com/contact#me>
   <http://www.w3.org/.../22-rdf-syntax-ns#type>
      <http://www.w3.org/.../contact#Person> .
```

The system needs to perform one last step. It detects that the domain identifier extracted from the subject URI differs from the domain identifier used to store the triples. It therefore needs to create the following reference triple and store it in the domain `http://example.com`.

```
<http://example.com/contact#me>
   <http://example.com/terms/usedInDomain>
      <http://www.w3.org> .
```

Now that the information has been stored in the DHT, it can be queried. A query consists of query path expressions, which must provide at least one constant value. If the constant value can not be used to extract a domain identifier, which can be the case if the only constant is a literal object, then the user has to provide the domain identifier explicitly. The following query is automatically created and executed to look-up if any relevant reference triples are stored in the domain:

```
select REFERENCE
from <[URI]> ns:usedInDomain {REFERENCE}
using namespace ns=&http://example.com/terms/
```

The placeholder *[URI]* is replaced with the URI of the constant value. The system then checks this domain, and any other domains referred to by reference triples found in the previous step, for intermediate results. This procedure is repeated until all query path expressions have been evaluated. Finally, the results are returned to the node that issued the query.

4.2 The Routing Table

The routing table manages two adjacency-lists, one for nodes that are in the same domain and one for nodes in other domains. They are accordingly referred to as the near and the far routing tables. This enables the routing table to easily differentiate between nodes that are close in terms of the DHT namespace and nodes that are further away. Nodes that are closer are more likely to be the target of messages routed by this node than nodes that are further away.

A node always forwards a message on a best effort basis. If the destination node is in one of the routing tables, then it can deliver it directly. Otherwise, it determines if the receiver is in the same domain or in a different domain. If the latter is the case, then it uses the far routing table to send the message to a node as close to the destination domain as possible. If, however, the recipient is in the same domain, then it uses the near routing table to determine the next hop. If none of the neighbours is closer to the destination node than the node itself, then it uses its leaf set to determine the next hop. The leaf set guarantees that a message can always be routed closer to its destination node, assuming that the leaf set is consistent and up-to-date. We use a leaf set size of 8 (4 on either side of the node) for the simulations. The ratio of the number of nodes in either one of the routing tables can be adjusted. Section 5.1 analyses how different ratios influence the routing performance.

Another configuration option is to specify the optimal neighbour distributions for the near and far routing tables. The routing table then attempts to find the entries for the adjacency-lists based on these distributions. For example, it is possible to specify that neighbours closer to the node itself, in terms of the domain namespace for the near routing table and in terms of the global namespace for the far routing table, are to be preferred. The distributions are expressed as lists of integers and must contain an odd number of elements. The distribution $\{1,2,3,2,1\}$ means that the respective namespace is split up into 5 segments. The local node always considers itself to be in the centre of the namespace. This is possible because the namespaces are viewed as rings, with their own upper and lower boundaries connected. In this example, the implementation would attempt to fill the respective routing table with $\frac{3}{9}$ of the neighbours from the middle segment of the namespace as seen from the node, $\frac{2}{9}$ of the neighbours from the next namespace segments on either side and $\frac{1}{9}$ of the neighbours from the outermost namespace segments. The following sections analyse how different routing table distributions affect the routing performance and network structure.

5 Network Analysis

We have developed a simulator that allows us to analyse the performance of the routing table in a large scale environment. We have analysed the generated network instances and the performance of the system in a number of ways. The results are presented in this section.

5.1 Properties of the Routing Table

Figure 2 shows how well different ratios of the near and far routing table sizes work in terms of the mean number of hops needed to deliver a message to its destination node. It also compares the three distinct neighbour distributions $\{1\}$ (a), $\{2,5,10,15,20,15,10,5,2\}$ (b) and $\{20,15,10,5,2,5,10,15,20\}$ (c). The results for distribution (a) and (b) do not vary much for the different routing table size ratios when the destination node is randomly chosen from all nodes in the network, whereas the results for distribution (c) are much more noisy and less consistent. This is not surprising, since the routing implementation expects an increased density of neighbour connections between nodes the closer they are to each other in the namespace of the DHT. Therefore, it was to be expected that distribution (b) performs better than (c) with (a) being somewhere in between. The stronger deviations from the mean values in the results of distribution (c) can also be found in the graphs described in the following sections.

Also not surprising is that an increase in the near routing table size leads to a steady decrease in the number of hops needed to deliver a message to the same domain as the source node. The far routing table is not used at all in this scenario, thus, a larger near routing table equals a bigger pool of relevant connections.

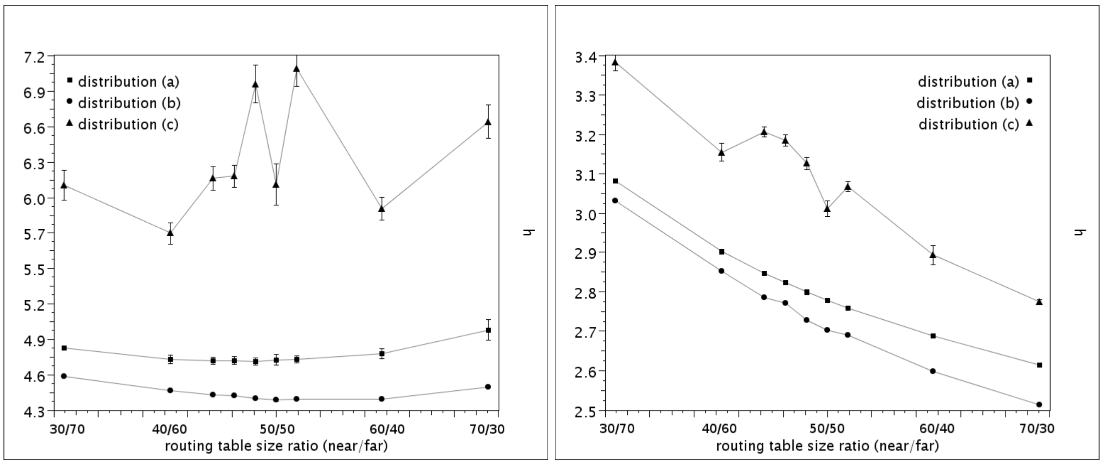

Figure 2: The graph on the left shows the mean number of hops h needed to deliver a message to a randomly chosen destination node, whereas the destination nodes for the graph on the right are chosen from the same domain as the source node. The results were measured for varying near and far routing table size ratios and the three distinct neighbour distributions described in the main text. The combined routing table size is always $k_{near} + k_{far} = k = 100$ and the network size is $N = 100,000$. The nodes are approximately evenly spread over $d = 100$ domains and $m \sim 1,600,000$ messages were generated per simulation for each destination node selection method. The results are averaged over 40 simulation runs; the error bars represent the standard deviations.

The results for distribution (c) look rather strange, but are most likely deceptive due to the increased number of data points between the ratios 40/60 and 60/40. We expect that further simulations will reveal that the lines in the lower and upper areas would not be as straight either if more data points were available.

5.2 Domains

So far the number of domains was fixed and the nodes were evenly spread over all the domains. Here we analyse how the domain size affects the results. Figure 3 shows the simulation results for networks with a constant number of nodes but with an increasing number of domains. More domains means smaller domains, and therefore the results for delivering messages within the same domain are not further surprising. When the destination nodes are selected randomly from all the nodes in the network, then the choice of the right domain size is more interesting. The results show that the introduction of domains had a positive effect, very few domains perform poorly. However, too many small domains do not perform good either, especially when the size of the far routing table is small. It is thus important to find a good balance for the given network size. In this case, a domain size of approximately 1,000 vertices ($d \approx 100$) appears to be a good value.

5.3 Scaling

As stated in section 3.1, scalability is an important property for our system. Figure 4 shows how the implementation deals with networks ranging from $1,000$ to $101,000$ nodes. At size $N = 1,000$, the results are the same for the two destination node selection methods, because the domain size is set to "1,000" and thus only one domain exists in the networks generated with these properties. Even when selecting the destination node randomly from all the nodes in the network, it merely takes ≈ 4.5 hops on average to deliver the message for the largest network generated so far. The total network size does not affect the delivery time for messages sent within the same domain, which shows how useful it can be to arrange the nodes in a way that most of the messages they send are addressed to their own domain. The result values

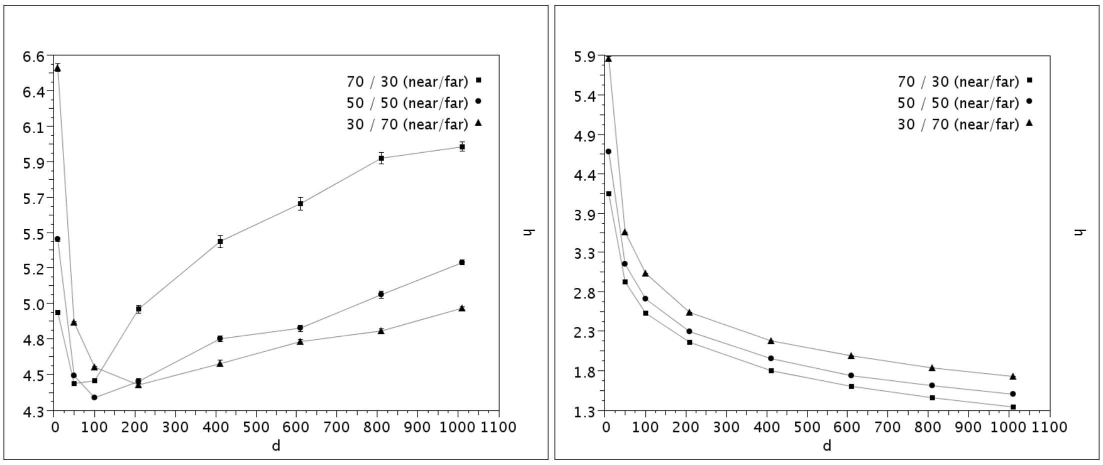

Figure 3: The two graphs illustrate how the number of hops h varies with the number of domains d in the network (ranging from $d = \{10, \ldots, 1010\}$) for three routing table size ratios. The destination nodes were chosen randomly from the whole network for the plot on the left, whereas they were chosen from the same domain as the source node for the plot on the right. The network size $N = 100,000$ is constant, and therefore the number of nodes in each domain decreases with increasing d. The degree $k = 100$ and the neighbours are selected according to distribution (b). Roughly $m \sim 1,600,000$ messages were generated per simulation for each destination node selection method. The results are averaged over 100 simulation runs; the error bars represent the standard deviations and are too small to be visible for some of the data points.

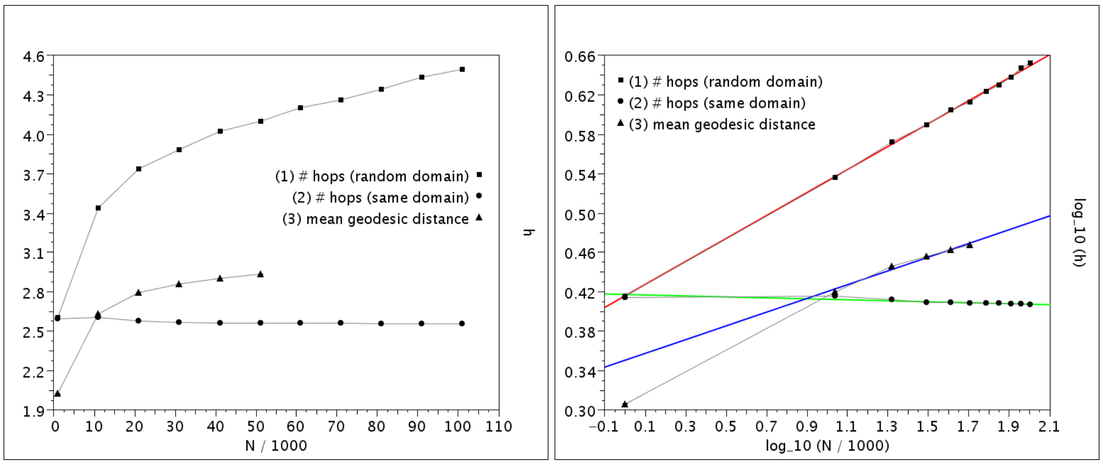

Figure 4: The graphs show the mean number of hops h needed to deliver a message to a destination node either randomly chosen (1) or from the same domain as the source node (2) for increasing network size ($N = \{1000, \ldots, 101000\}$). (3) shows the mean geodesic distance, that is the mean shortest path length between any two vertices in the network for networks up to 51000 nodes. The graph on the right shows the data on a log-log scale. The straight lines are the least square linear fits with a slope of ≈ 0.117 for (1), ≈ -0.005 for (2) and ≈ 0.070 for (3). The number of nodes in each domain is always $1,000$, and therefore the number of domains $d = \{1, 11, \ldots, 101\}$ increases with the network size. The degree $k = 100$ and the neighbours are selected according to distribution (b). The routing table size ratio is 70/30 for the near/far routing tables. Roughly $m \sim N * 16$ messages were generated per simulation for each of the destination node selection methods. The results are averaged over 100 simulation runs for (1) and (2) and over 40 network instances for (3); the error bars representing the standard deviations are too small to be visible.

even decrease slightly for (2) with increasing network size. The most likely explanation for this is that, even though the number of messages increases linearly with the network size, more messages are routed through the network in total, and this means that the system has more opportunities to optimise the routing tables.

The plot also shows the mean geodesic distances for the network instances (3). These are the mean shortest paths between any two nodes in the networks, and thus the absolute minimum number of hops, on average, needed to route a message from a source node to its destination in the given network structure. This is an expensive metric to compute and, even though we were utilising the massively parallel processing power of an up-to-date GPU (Graphics Processing Unit) [54], we were only able to compute it for networks with up to $51,000$ nodes within the time given to us.

One of the characteristic properties of small-world networks, as stated in section 3, is that the mean geodesic distance scales logarithmically or slower with the network size for fixed degree k. The right plot in figure 4 shows that, as far the data available to us suggests, our network model fulfils this requirement both for the geodesic distance and for the actual hops needed to route messages to randomly chosen nodes.

5.4 Clustering

As mentioned in section 3, small-world networks show larger clustering coefficients than random graphs. We use two different definitions to determine the clustering behaviour of the networks. Figure 5 shows the results for these clustering coefficients, each for varying routing table size ratios and neighbour distributions. The graph on the left shows the clustering coefficient γ as defined by Watts and Strogatz [46, 55]. The value γ of a graph G is γ_v averaged over all $v \in V(G)$. γ_v is defined as follows:

> The clustering coefficient γ_v of Γ_v characterises the extend to which vertices adjacent to any vertex v are adjacent to each other. More precisely,
>
> $$\gamma_v = \frac{|E(\Gamma_v)|}{\binom{k_v}{2}}, \tag{1}$$
>
> where $|E(\Gamma_v)|$ is the number of edges in the neighbourhood of v and $\binom{k_v}{2}$ is the total number of *possible* edges in Γ_v.

As expected, networks generated with distribution (c) have a lower clustering coefficient than the other networks. The values for distribution (a) and (b) are more interesting. When the routing table size ratio is in favour of the far routing table, then distribution (b) yields slightly higher results than distribution (a), which is what we expected to be the case. However, once the ratio changes in favour of the near routing table, distribution (a) shows a clearly higher clustering than distribution (b). It is also not clear why the clustering coefficients for all distributions decrease the more the routing table size ratio favours the near routing table, until they begin to increase again more or less significantly around the time the near routing table is actually larger than the far routing table. We expected that a proportionally larger near routing table, which means more neighbours are chosen from the same domain and thus from closer to the source node in terms of the namespace, would yield larger clustering coefficients.

The graph on the right of figure 5 shows the clustering profile [56]. The clustering profile $C^d(v)$ for a vertex $v \in V$ was originally defined as the number of neighbours of v connected by a shortest path of length d that does not go through v, divided by the total number of pairs of neighbours of v. Or in other words, $C^d(v)$ is the number of neighbours of v whose smallest cycle shared with it has length $d+2$, divided by the total number of pairs of neighbours of v.

However, this definition only works for undirected graphs. In the case of directed graphs, like the ones in our simulations, a vertex u can have v in its adjacency-list without itself being in the adjacency-list of v. Thus, we modified the definition to work with directed graphs as the one in figure 6. The number of cycles of length $d+2$ that contain v and the number of paths of length $d+1$ that start from v (a vertex may only appear once in each cycle and path) are counted. The fraction of vertices at distance $d+1$ from v that have v in their adjacency-list—and are thus closing a cycle—is the clustering profile $C^d(v)$.

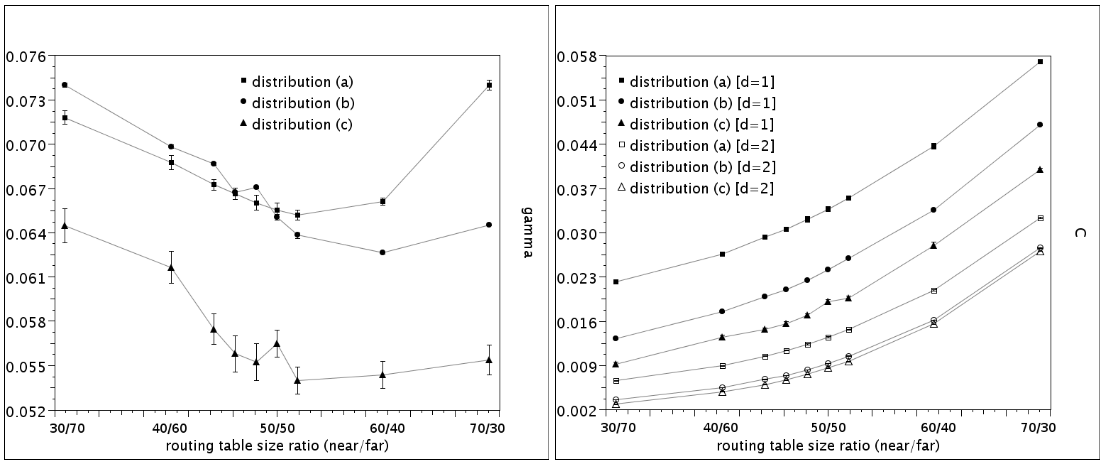

Figure 5: The graphs show the results for two different definitions of the clustering coefficient for varying routing table size ratios and the three distinct neighbour distributions. The results γ calculated with Watts' clustering coefficient are illustrated by the plot on the left and the clustering profile C by the plot on the right. The combined routing table size is always $k_{near} + k_{far} = k = 100$ and the network size is $N = 100,000$. The nodes are approximately evenly spread over $d = 100$ domains. The results are averaged over 40 networks; the error bars representing the standard deviations are too small to be visible for some of the data points.

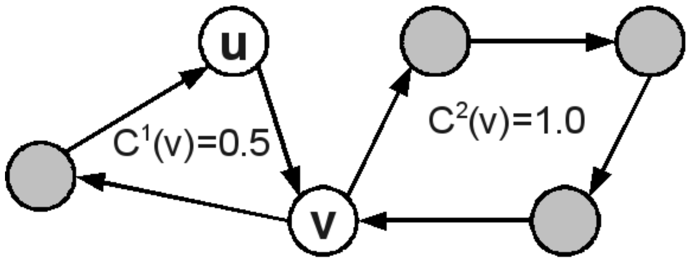

Figure 6: Higher orders of clustering in a directed graph G. The clustering profiles for G by degree C_k^d are $C_1^1 = 0.3$, $C_2^1 = 0.5$, $C_1^2 = 0.4$ and $C_2^2 = 1.0$. The total clustering profiles averaged over all vertices are $C^1 \approx 0.33$ and $C^2 = 0.5$.

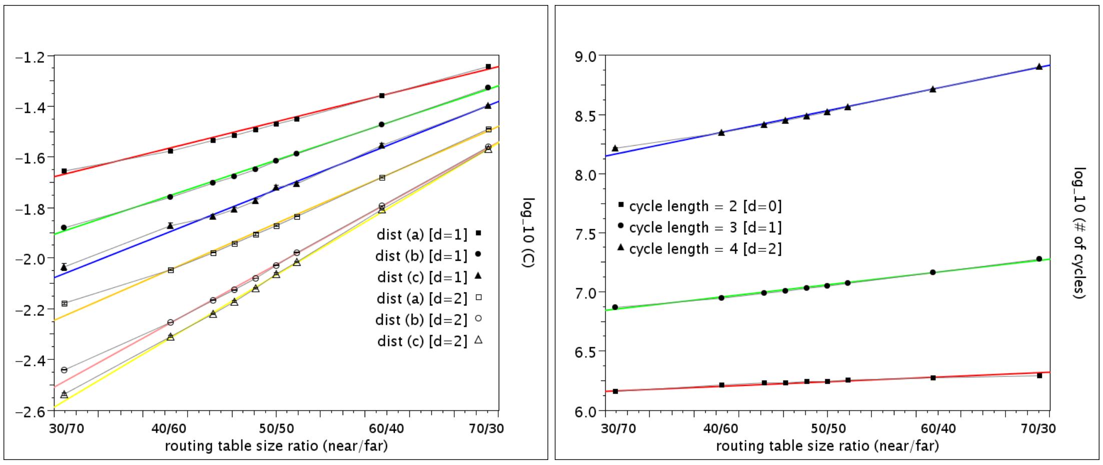

Figure 7: The graph on the left shows the results for the clustering profile (see figure 5) on a log-y scale. The straight lines are the least square linear fits. The slopes (from top to bottom) are $\approx \{0.010, 0.014, 0.017, 0.018, 0.023, 0.025\}$. The graph on the right shows the number of cycles of length 2, 3 and 4 found in the networks generated with distribution (a). It is also on a log-y scale with the least square linear fits displayed. The slope for cycle length 2 is ≈ 0.004, for length 3 it is ≈ 0.010 and for length 4 it is ≈ 0.018.

$$C^d(v) = \frac{\text{cycles of length } d+2 \text{ containing } v}{\text{paths of length } d+1 \text{ starting from } v} \tag{2}$$

In this definition, the clustering profile with $d = 1$ is the same as the clustering coefficient defined by Newman [57, 38] and can thus be considered an extended version of it.

The clustering profile can be used to gain more information about the network than the definition by Watts, as it makes it possible to determine a more in depth clustering behaviour. Unfortunately, with networks of size $N = 100,000$ vertices and degree $k = 100$, the number of cycles of a certain length increases dramatically (see the graph on the right hand side of figure 7) with every increase in cycle length, restricting us to cycles of up to length 4 only.

The results given by this second definition look much more like what we expected to see. They consistently increase with the routing table size ratio changing in favour of the near routing table. But like in the previous results, distribution (a) again yields the largest values for the clustering profile and not distribution (b).

The left graph in figure 7 shows the results of the clustering profile on a log-y scale. The roughly straight lines indicate that the clustering profile grows almost exponentially with the routing table size ratio leaning towards the near routing table. This does not mean that this trend continues outside the measured ratios though. It has to be kept in mind that there is a hard limit at the ratios 0/100 and 100/0, as this would mean that one of the routing tables is completely empty.

6 Discussion & Future Directions

We have analysed a variety of metrics relevant to a distributed peer-to-peer system and small-world graphs, using different configuration sets to show their effects on the system performance. We have not yet analysed the network robustness, which is another important property of a reliable system.

We are planning to perform additional simulations to gain further insights into the network structure and the effects that different parameter values have. For example, so far the routing table size has always been constant (degree $k = 100$). It will be interesting to see how the system performance changes with smaller and larger degrees.

The neighbour selection algorithm does not only take the neighbour distribution, but also the latency between the nodes into account when it chooses a new neighbour. This is an important feature to improve the performance in a real-world deployment scenario. However, this has not been incorporated into the network analysis yet. It will be interesting to see how the neighbour "quality" changes over time, how this affects the mean shortest paths over time when the arcs are weighted with the latency and how the actual message delivery times change.

But not only further simulations and analyses are planned, improvements and experimental features to the routing table implementation are also on our to-do list:

- Preferential attachment to increase the cluster formation in the network. The probability to add a node as a neighbour increases with the number of current neighbours that already have that node in their routing tables.
- Preferential attachment based on the number of messages routed to a certain node. That is, increase the probability to add a new neighbour based on how often it is the destination of messages generated by or routed through the local node.
- Dynamic assignment of nodes to domains. In the current incarnation, the domain has to be specified when a node is started. The dynamic re-assignment of a node to a certain domain based on the node's usage of the system would add more flexibility.

We described the storage and querying algorithms for RDF metadata in section 4, but so far we have only concentrated on the analysis of the underlying network structure and have neglected to analyse the performance of these algorithms. This is another task on our to-do list.

7 Conclusions

We have given an overview over previously published resource discovery material and described our own prototype system for the distributed peer-to-peer storage of semantic metadata. The underlying network model is based on a Distributed Hash Table that was modified to create a network that possesses the characteristic properties of small-world graphs. Especially the scaling behaviour of the small-world network model is important for a resource discovery system that is supposed to be able to handle large scale deployments of many thousands or even millions of nodes. We introduced the concept of domains to store related information. Messages that are sent between nodes in the same domain are typically delivered within less hops than messages sent to an arbitrary destination node and the number of hops grows with the domain size and not with the network size. We analysed a variety of network metrics in a simulated environment and showed that it does indeed scale logarithmically or slower with the network size. It seems likely that e-scientific user communities will continue to organise themselves into small-world structures that scale in this way.

The global e-Science experiment is still ongoing, but e-Science will likely just become a standard part of the infrastructure for "doing science". Although the resource discovery problem for support of a user-transparent e-Science infrastructure is not yet completely solved, we believe the approaches we have discussed will go a long way towards a practical solution.

References

[1] Fox, G.C., Williams, R.D., Messina, P.C.: Parallel Computing Works! Morgan Kaufmann Publishers, Inc. (1994) ISBN 1-55860-253-4.

[2] Kirkham, D.: Telstra's experimental broadband network. Telecommunications J. Australia **45** (1995)

[3] Hawick, K.A., James, H.A., Silis, A.J., Grove, D.A., Kerry, K.E., Mathew, J.A., Coddington, P.D., Patten, C.J., Hercus, J.F., Vaughan, F.A.: DISCWorld: An environment for service-based metacomputing. Future Generation Computing Systems (FGCS) **15** (1999) 623

[4] Foster, I., Kesselman, C., eds.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, Inc. (1999) ISBN 1-55860-475-8.

[5] Buyya, R., Yeo, C.S., Venugopal, S.: Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. In: Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008). (2008) 5–13

[6] Milgram, S.: The Small-World Problem. Psychology Today **1** (1967) 61–67

[7] Hawick, K., James, H.: Managing community membership information in a small-world grid. Research Letters in the Information and Mathematical Sciences **7** (2005) 101–115 ISSN 1175-2777.

[8] MOA Collaboration: Microlensing Observations in Astrophysics (MOA) Collaboration. `http://www.phys.canterbury.ac.nz/moa/` (2009) Last accessed January 2009.

[9] Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., Tuecke, S.: A resource management architecture for metacomputing systems. In Feitelson, D.G., Rudolph, L., eds.: JSSPP'98. Number 1459 in LNCS, Springer-Verlag Berlin Heidelberg (1998) 62–82

[10] Rana, O.F., Bunford-Jones, D., Hawick, K.A., Walker, D.W., Addis, M., Surridge, M.: Resource discovery for dynamic clusters in computational grids. In: Proceedings of the 15th International Parallel & Distributed Processing Symposium. (2001) 82

[11] W3C: RDF Primer. (2004) Last accessed August 2008.

[12] W3C: RDF Vocabulary Description Language 1.0: RDF Schema. (2004) Last accessed August 2008.

[13] W3C: RDF/XML Syntax Specification (Revised). (2004) Last accessed August 2008.

[14] Hauswirth, M., Schmidt, R.: An overlay network for resource discovery in grids. In: Proc Sixteenth International Workshop on Database and Expert Systems Applications. (2005) 343– 348 ISBN: 0-7695-2424-9.

[15] Rana, O.F., Bunford-Jones, D., Walker, D.W., Addis, M., Surridge, M., Hawick, K.: Agent-based resource discovery for dynamic clusters. In: Proc. Second IEEE International Conference on Cluster Computing (CLUSTER'00), Chemnitz, Germany (2000) 353 ISBN: 0-7695-0896-0.

[16] Hawick, K.A., James, H.A.: Modeling a gossip protocol for resource discovery in distributed systems. In: Proc. Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA 2001), Las Vegas, USA (2001) DHPC-102.

[17] Mockapatris, P., Dunlap, K.: Development of the domain name system. ACM Computer Communications Review **18** (1988) 123–133

[18] von Laszewski, G., Foster, I.: Usage of ldap in globus. Available from http://www.globus.org (1998)

[19] Foster, I., Kesselman, C.: Globus: A meta-computing infra-structure toolkit. Int. J. Supercomputer Applications (1996)

[20] Atlas Project: Atlas Distributed RDF Metadata Storage and Query System. `http://atlas.di.uoa.gr` (2008) Last accessed April 2008.

[21] Liarou, E., Idreos, S., Koubarakis, M.: Evaluating Conjunctive Triple Pattern Queries over Large Structured Overlay Networks. In: Proceedings of the Semantic Web - ISWC 2006: 5th International Semantic Web Conference, Athens, GA, Springer-Verlag (2006) 399–413

[22] Hawick, K., James, H.: Simulating resource trading and discovery in grid systems. In: IASTEd Conference on Modelling, Simulation and Optimisation (MSO 2003), Banff, Canada, IASTED (2003)

[23] Iamnitchi, A., Foster, I.: A peer-to-peer approach to resource location in grid environments. In Nabrzyski, J., Schopf, J., Weglarz, J., eds.: Grid Resource Management: State of the Art and Future Trends. Volume 64 of International Series In Operations Research & Management Science. Kluwer Academic Publishers (2003) 413–429

[24] Al-Dmour, N., Teahan, W.: Peer-to-peer protocols for resource discovery in the grid. In: IASTED International Conference on Parallel and Distributed Computing and Networks, Innsbruck, Austria, IASTED (2005)

[25] Chunlin, L., Layuan, L.: Agent framework to support the computational grid. The Journal of Systems and Software **70** (2004) 117–187

[26] Ding, S., Yuan, J., Ju, J., Hu, L.: A heuristic algorithm for agent-based grid resource discovery. In: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05), IEEE (2005) 222–225 ISBN:0-7695-2274-2.

[27] Ranjan, R., Chan, L., Harwood, A., Karunasekera, S., Buyya, R.: Decentralised resource discovery service for large scale federated grids. In: Proc. Third IEEE International Conference on e-Science and Grid Computing. (2007) 379–387

[28] Iamnitchi, A., Foster, I.T.: On fully decentralized resource discovery in grid environments. In: Proceedings of the Second International Workshop on Grid Computing, London, UK., Springer-Verlag (2001) 51–62

[29] Iamnitchi, A., Foster, I., Nurmi, D.C.: A peer-to-peer approach to resource discovery in grid environments. In: High Performance Distributed Computing, IEEE (2002)

[30] Mastroianni, C., Talia, D., Verta, O.: A super-peer model for resource discovery services in large-scale grids. Future Generation Computer Systems **21** (2005) 1235–1248

[31] Naseer, A., Stergioulas, L.K.: Resource discovery in grids and other distributed environments: States of the art. Multiagent and Grid Systems - An International Journal **2** (2006) 163–182

[32] Filali, I., Huet, F., Vergoni, C.: A simple cache based mechanism for peer to peer resource discovery in grid environments. In: Proc. Eighth IEEE International Symposium on Cluster Computing and the Grid, IEEE (2008) 602–608

[33] Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling Churn in a DHT. In: Proceedings of the General Track 2004 USENIX Annual Technical Conference, Boston, MD, USENIX Association, Berkeley (2004) 127–140

[34] Jeong, H., Tombor, B., Albert, R., Oltvai, Z., Barabsi, A.L.: The large-scale organization of metabolic networks. Nature **407** (2000) 651–654

[35] Fell, D.A., Wagner, A.: The small world of metabolism. Nature Biotechnology **18** (2000) 1121–1122

[36] Wagner, A., Fell, D.A.: The small world inside large metabolic networks. Proceedings of the Royal Society B **268** (2001) 1803–1810

[37] Newman, M.E.J.: The structure of scientific collaboration networks. PNAS **98** (2001) 404–409

[38] Newman, M.E.J.: Ego-centered networks and the ripple effect. Social Networks **25** (2003) 83–95

[39] Xu, J., Chen, H.: The Topology of Dark Networks. Communications of the ACM **51** (2008) 58–65

[40] Liljeros, F., Edling, C., Amaral, L., Stanley, H., Aberg, Y.: The web of human sexual contacts. Nature **411** (2001) 907–908

[41] Liben-Nowell, D., Kleinberg, J.: Tracing information flow on a global scale using Internet chain-letter data. PNAS **105** (2008) 4633–4638

[42] Albert, R., Jeong, H., Barabsi, A.L.: Diameter of the World-Wide Web. Nature **401** (1999) 130–131

[43] Barabsi, A.L., Albert, R.: Emergence of scaling in random networks. Science **286** (1999) 509–512

[44] Price, D.J.d.: Networks of Scientific Papers. Science **149** (1965) 510–515

[45] Newman, M.E.J.: The Structure and Function of Complex Networks. SIAM Review **45** (2003) 167–256

[46] Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature **393** (1998) 440–442

[47] Bamboo Project: Bamboo Distributed Hash Table. `http://www.bamboo-dht.org` (2008) Last accessed April 2008.

[48] Albert, R., Jeong, H., Barabsi, A.L.: Error and attack tolerance of complex networks. Nature **406** (2000) 378–382

[49] Holme, P., Kim, B.J., Yoon, C.N., Han, S.K.: Attack vulnerability of complex networks. Physical Review E **65** (2002)

[50] Buchanan, M.: NEXUS - Small Worlds and the Groundbreaking Science of Networks. W.W. Norton & Company (2002)

[51] Leist, A., Hawick, K.: A Small-World Network Model for Distributed Storage of Semantic Metadata. In Kelly, W., Roe, P., eds.: Proc. 7th Australasian Symposium on Grid Computing and e-Research (AUSGRID 2009). Volume 99 of Conferences in Research and Practice in Information Technology (CRPIT). (2009)

[52] Watts, D.J.: Six Degrees - The Science of a Connected Age. W.W. Norton & Company (2003)

[53] W3C: Resource Description Framework (RDF): Concepts and Abstract Syntax. (2004) Last accessed August 2008.

[54] Leist, A., Playne, D., Hawick, K.: Exploiting Graphical Processing Units for Data-Parallel Scientific Applications. Concurrency and Computation: Practice and Experience **21** (2009) 2400–2437 CSTN-065.

[55] Watts, D.J.: Small Worlds: The Structure of Networks between Order and Randomness. Princeton University Press (1999)

[56] Abdo, A.H., de Moura, A.P.S.: Clustering as a measure of the local topology of networks. Technical report, Universidade de São Paulo, University of Aberdeen (2008)

[57] Newman, M.E.J., Strogatz, S.H., Watts, D.J.: Random graphs with arbitrary degree distributions and their applications. Physical Review E **64** (2001) 026118