[DMSt04] *Vasilios Darlagiannis, Andreas Mauthe, Ralf Steinmetz;* **Overlay Design Mechanisms for Heterogeneous, Large Scale, Dynamic P2P Systems;** Journal of Network and Systems Management, Special Issue on Distributed Management, 12(3), August 2004, S. .....

Overlay Design Mechanisms for Heterogeneous, Large Scale, Dynamic P2P Systems

Vasilios Darlagiannis, Andreas Mauthe, Ralf Steinmetz
Darmstadt University of Technology
Multimedia Communications (KOM)
Merckstr. 25, 64283 Darmstadt, Germany
{Vasilios.Darlagiannis, Andreas.Mauthe, Ralf.Steinmetz}@KOM.tu-darmstadt.de

ABSTRACT Large scale, heterogeneous Peer-to-Peer (P2P) systems impose a set of diverse requirements. Current solutions do commonly only address a subset of these requirements since there is a number of trade-offs and constraints due to the different dimensions and aims they address.

In this paper we present a novel approach for designing overlay networks for large scale, highly dynamic and heterogeneous P2P systems. A set of mechanisms is proposed to meet the complete set of requirements while keeping the trade-offs and constraints in balance. In order to handle effectively the large number of peers, they are clustered in manageable groups considering the requirements on their stability. The novelty in this approach is in the identification of the core services and operations of the aforementioned systems. Based on the requirements of those services and operations, peers are assigned the most suitable roles. Role relationships are further introduced to enable (and provide) incentives for the peers to adopt the most suitable roles while selecting an efficient overlay structure to preserve efficiency, robustness and scalability.

The proposed set of mechanisms is realized in Omicron, a novel hybrid P2P approach.

KEYWORDS Peer-to-Peer Systems, Peer Roles, Heterogeneity, Stable Clusters, Load Balance.

1 INTRODUCTION

P2P systems have been receiving considerable attention from the networking research community recently. A number of approaches have been proposed as communication schemes in order to provide efficient and scalable inter-peer communication. These schemes are designed on top of the physical networking infrastructure as *overlay networks.* Their topologies influence greatly the performance of the routing algorithms and hence the efficiency of a P2P system.

There is a set of requirements that has to be met to design overlay networks that are easily deployed on top of the Internet while making use of available resources optimally. Present overlay design approaches conventionally concentrate on a subset of them and ignore the importance of others. Such approaches are usually also inflexible in adapting to emerging requirements in a later step without compromising the already fulfilled ones. Complete redesign is usually the only solution.

This can be demonstrated by briefly analyzing some well-known proposals. Gnutella [1] for instance offers a non-hierarchical P2P approach with minimum maintenance cost. However, the employed flooding mechanism used for querying make Gnutella unscalable and caused a system breakdown. Learning from the Gnutella approach two main directions have been followed by the P2P overlay network designers.

One is the redesign of P2P systems using hierarchical architectures. However, it is questionable whether they can still be called P2P. In these systems the concept of super-peers is introduced (i.e. peers with additional capabilities). The super-peers usually form the backbone of the overlays having the normal peers placed around them. As typical representatives of this design direction the architectures of eDonkey [2] and KaZaa [3] can be considered. The basic drawback of this approach is the requirement for the existence of super-peers, which subsequently imposes new requirements on the system. Super-peers have to operate correctly since their actions have greater effect in every overlay operation. Further, a malicious behavior of these peers has far greater impact than in the case of non-hierarchical P2P systems. Super-peer failures are more severe than normal peer failures. Additionally, there is lack of incentives for users to serve the rest of peers. A super-peer can become also a performance bottlenecks if there is not a sufficient number of them.

The second approach is based on Distributed Hash Tables (DHTs). As their name implies, they are distributed structures that use hash functions to index items that should be located on the overlays. Identities are assigned to peers that take values from the same key space as the indexed items. A minimum distance function is usually used to map items to responsible peers. DHT-based systems aim to provide very efficient routing of messages to their logical destinations. The cost of the routing mechanism grows logarithmically with respect to the size of the overlay network for most of the proposed approaches. This applies to the number of hops required for routing as well as the size of the routing tables. Typical representatives of this approach are Chord [4], CAN [5] and Pastry [6]. The increased overhead for maintaining the structure of these systems is a significant cost factor. This can become inefficient in scenarios where high peer join/leave rates occur. Moreover, each node is assumed to have a relatively big number of connections (for large-size systems) that grows logarithmically with the system size. Although this requirement increases the robustness of the overlay network, it is hard to be fulfilled by nodes with low physical capabilities.

Some other systems follow hybrid approaches. Brocade [7] and JXTA [8] are characteristic examples. Their architecture aims to take advantages of both hierarchical and DHT-based solutions. SHARK [9] is another approach that combines structured and unstructured mechanisms and correlates peers with similar interests to form unstructured groups that can be reached through the structured part of the overlay.

In order to overcome the large maintenance cost caused by the high churn rate of peers in the overlay and the logarithmically increased connectivity size (with respect to the network size) some additional approaches were proposed. Koorde [10] (a variation of Chord) employs de Bruijn graphs [11]. The advantage of this approach is the constant number of connections per peer. Though, this comes at the cost of decreased robustness and a difficulty in incrementally extending the network size. A logarithmic connectivity-based version of Koorde overcomes the robustness problem and provides optimal lookup operations. However, it comes at the cost of high maintenance overhead. A different approach, named Viceroy [12], is based on the butterfly graph. Viceroy requires only a constant number of neighbors with high probability. Though, the construction and maintenance procedures are relatively complex.

The approach introduced in this paper aims to provide general *Overlay Design Mechanisms* for hybrid, DHT-based approaches. The rational in our approach is to reduce the high maintenance cost by having a small, constant number of connections and routing table sizes (at least for the majority of the peers). For this reason the usage of appropriate graph structures (such as de Bruijn or Kautz [13]) is suggested. These graphs are also called lexicographic. Since this solution causes robustness problems clusters of peers are formed with certain requirements on their stability. The de Bruijn overlay is used for inter-cluster connectivity. Going a step further a role-based scheme is introduced to deal with the heterogeneity of the peer capabilities and user behavior. This scheme fits the contribution of each node to its capabilities and aims at the maximization of the cluster efficiency by providing appropriate incentives for each role.

The rest of this paper is organized as follows. In the next section the major requirements of large scale, highly dynamic and heterogeneous overlay networks and their cross-effects are identified. Subsequently, a set of core mechanisms to meet these requirements is proposed in Section 3 that addresses their potential side effects. In Section 4 additional mechanisms and algorithms to handle effectively the management of the overlay are provided. Section 5 illustrates the feasibility of our solution by describing a novel overlay network infrastructure called Omicron. Related work is described in Section 6. Finally the paper is concluded in Section 7.

2 REQUIREMENTS AND CROSS-EFFECTS

This section presents the most important requirements that should be considered in the design phase of the overlay. Potential cross-effects (that are caused by certain combinations of solution mechanisms) are outlined as well.

2.1 Requirements

The main task of designing a network overlay is constructing the graph that represents the topology characteristics (the nodes and their connectivity pattern). In order to provide an efficient solution the designer should also consider the core (basic) services and operations that will be deployed over this graph. Core services and operations are for instance the routing of messages or the maintenance of the desired graph structure. As an example of the close relationship between the graph structure and the operation of core services consider the (trivial) case of a graph that forms a star. In this topology the central node represents a so-called communication hot-spot and a central point of failure.

For this reason both static and dynamic characteristics of the graphs have to be considered. Further, the additional requirements that are imposed by the deployed core services of the overlay networks are examined. In this section the major set of requirements are introduced.

2.1.1 Scalability and incremental extendability.

A simple definition for *scalability* is the ease with which a system or component can be modified to fit the problem sphere. Scalability in the design of overlay networks can be measured in time and space dimensions. However, there is always a trade-off. Time complexity involves the number of hops that are required to forward the messages from the source node to the destination. Space complexity is related to the size of the state that each node is required to maintain in order to keep the overlay in a functioning state. This information includes the routing table entries of the neighbor nodes as well as the indexing structures for the advertised items. Additionally, the size of the messages and the overhead of the maintenance procedure should be additionally considered in order to evaluate the scalability of the overlay network.

Extendability of the topology (meaning the ease with which the topology graph can be extended to larger sizes) is also very important. It expresses how a system or a component can be modified to increase its storage, communication or functional capacity at low cost. The topology is incrementally extendable if the definition of the topology allows graphs of any size. Otherwise (if the size granularity is only greater than one) they are defined as *partially extendable*. Some hierarchically recursive topologies allow graphs of specific discrete sizes (such as powers of two). If a topology is incrementally extendable a very important question is how the structure of an instance of size n differs from the structure of an instance of size $n + k$ for some integer constant $k \geq 1$.

2.1.2 Fault-tolerance and stabilization.

Fault-tolerance is the ability of a system or component to continue normal operation despite the presence of hardware or software failures. In terms of overlay networks it expresses the resilience of the connectivity when failures are encountered by the arbitrary departure of peers[1].

P2P systems have been studied extensively [14], [15], [16]. One main aspect among others is the observation of user behavior. An interesting result was the fact that the majority of the peers tend to stay connected with the system for a relatively short time. This results in a highly dynamic system with high join and leave rates.

This kind of behavior imposes another requirement especially important for structured overlays. Effects like overlay partitioning are possible in case where a special overlay maintenance procedure is not in place. High maintenance costs to keep the structure in a *stable* state and provide reliable services are some of the consequences.

2.1.3 Support for heterogeneity.

Overlay network design should take into account the *heterogeneity in the physical capabilities of the peers and the user behavior.* Designing schemes that require homogenous components can either decrease the system capabilities to those achievable by the weakest components or faulty/inefficient operation should be expected from the least capable nodes.

Moreover, the observed variation in node behavior (e.g. up-time patterns) [17] should be taken into account in the design of the overlay to increase the efficiency of the systems.

2.1.4 Load Balance and fairness.

Load-balance is defined as the extent to which the load is evenly spread across nodes. The load considered in this context consists of the effort required for the basic overlay operations, e.g. maintenance, routing, indexing, caching, etc. Designing an overlay network that avoids hot spots increases the performance and the fault-tolerance of the overall system.

On the other hand, by taking into account the heterogeneous environment not all of the nodes are capable of offering the same amount of resources. A *fair* solution on the offered services should provide the incentives and the weighted balance between the resource contribution and the consumed overlay services.

2.1.5 Efficiency

Efficiency is defined as the ratio of productive resource consumption and total resource consumption. In the context of network overlays efficiency can be described in terms of *routing* and *physical network proximity.*

The actual packet routing from a source node to a destination node is a service offered by the underlying network. The complexity of this layer (caused by the way IP addresses are assigned) should not be repeated in the overlay network. Combined with an appropriate node identification scheme the overlay routing mechanism should be a simple operation. Based on the destination node address the mechanism should return the closest neighbor to the destination.

Network proximity (locality) consideration is a crucial factor in reducing the latency on the network operations. Relative Delay Penalty (RDP) is a measure of the additional packet delay introduced by the overlay on the delivery of a message between two nodes. It is defined as the ratio of the latency experienced when sending data using the overlay compared to the latency experienced when sending data directly using the underlying network. Link stress is another metric that quantifies the usage of the underlying overlay from the overlay. It is defined as the number of tunnels that send traffic over a physical link. Links can be "stressed" despite the fact

[1] We abstract from the physical network failures

that the overlay traffic is well-balanced among its nodes. Studies of the Gnutella system [18] observed that requests forwarding performed many "zig-zag" forwarding steps between nodes that are physically located in North America and Europe.

2.1.6 Additional Requirements

The aforementioned requirements are those the set of mechanisms in this paper focuses on. However, some additional requirements that are important for certain scenarios and use cases should not be ommited.

Autonomous nodes: P2P systems are composed of a set of independent nodes that are not centrally controlled or administered. Despite this fact many proposals adapted hierarchical solutions for efficiency reasons (e.g. KaZaa [3] and eDonkey [2]). Such systems imposes additional requirements on the supporting infrastructure. In the case of eDonkey particularly the servers run different software. This is an even stronger requirement than potentially assigning a super-peer role to certain capable peers. Flat approaches (either structured like Chord and Pastry, or unstructured like Gnutella and Freenet [19]) are definitely preferable over the hierarchical alternatives since they maintain peer autonomy.

Security: *Security* is the ability of a system to manage, protect and distribute sensitive information. In the context of P2P overlay networks security issues are basically raised by the presence of malicious peers.

Additionally, selfish peers can behave in a way that could have similar results. However, security issues are out of the scope of this paper.

Anonymity: *Anonymity* can be defined as the degree to which a system or component allows for (or supports) anonymous transactions. This is a special requirement for certain applications that require anti-censorship features or to increase the privacy of the participating users. Still, a node identification scheme is required but message delivery require modification of certain important header fields in order to prevent the extraction of identity information. Of course, misusage of those systems is always an issue. Anonymity is not directly addressed by the presented proposal.

2.2 Trade-offs and constraints

This section presents critical interference and effects between pairs of conflicting requirements from the above identified set of requirements. The degree at which the conflicting requirements can be fulfilled is discussed in detail.

Fault-tolerance versus heterogeneity. In the context of P2P systems where peers represent unreliable components fault-tolerance is achieved mostly by the use of redundancy and replication mechanisms. DHT-based approaches suggest a large number of neighbors that usually increases logarithmically with respect to size of the system. While it has been shown that such an approach provides high fault-tolerance [20] it ignores practical limitations raised by peers of low physical capabilities.

Scalability versus extendability. Certain topologies where the diameter increases logarithmically with respect to the size of the network have been proposed to serve as P2P interconnection networks. However, many of these topologies can not be defined for arbitrary network sizes. For instance hypercubes can be defined for network sizes that increment exponentially. This limitation is critical for P2P systems where participation cannot be controlled. Certain algorithms can modify the aforementioned topologies to make them incrementally expandable, though, the resulted topologies diverge from the optimal original one.

Heterogeneity versus load balance. Designing large-scale, self-organized systems can be a great challenge in scenarios where a large number of low capability and unstable peers participate. Currently, only non-autonomous systems (e.g. KaZaa, eDonkey) seem to work efficiently

in wide deployments (with the exception of Overnet[2] [22]). Following such hierarchical solutions the workload is unevenly distributed among normal peers and super-peers. A fair, incentive-based mechanism is hardly achievable in services where micro-payment based solutions add a great overhead. On the other hand, following non-hierarchical solutions needs to go beyond the currently proposed schemes to achieve efficiently fault-tolerant and stable overlays. Adaptive mechanisms are required to provide the maximal efficiency while preserving the autonomy of the peers.

Anonymity versus security. Full anonymity in every aspect of the transactions and high security levels are by themselves extremely challenging topics and are currently being investigated in detail. Their combination is an even harder problem since in many cases designers have to decide for either alternative. In a fully anonymous environment, malicious peers can act freely while in a highly secure environment, strong identification is usually a requirement. Additionally, the identification scheme can set the level of anonymity and user privacy. Taking again the example of using native IP addresses, actions can be mapped to users in a more direct way. On the other hand, an appropriately selected identification scheme can be combined with *cryptographic* techniques to increase the level of security in the system. It should be noted that although the proposed set of design mechanisms does not address directly security and anonymity issues they are critical requirements for many applications.

3 MECHANISMS

Before the presentation of the core mechanisms for network overlay design it is necessary to describe the basic components that are required for their realization.

Routing tables: The *routing tables* are vital components of the routing scheme. They should be efficiently structured to be able to represent peer's local view of the graph topology. Also, they should be sufficiently powerful to augment the efficient routing of every message independently of the destination and despite of the limited local view.

Indexing structures: Although no details on the *indexing scheme* employed to provide information about the advertised items are described in this paper indexing structures are very important. They have to be designed in a way that allows to re-distribute and re-organize them at low cost. Peers join and leave the system dynamically so there can be frequent re-arrangements of the indexing structures.

Identification scheme: One of the first tasks in designing an overlay network is the selection of an appropriate *identification scheme*. Entities that require identification are first of all the nodes. However, potentially more components might need identification, such as clusters, resources, etc. Usually the identifiers are randomly chosen from a large key space. This selection can have a great impact on other design requirements (namely for the efficient routing mechanism, the efficient distribution of load, the security and the anonymity). Advanced mechanisms might be required for the identification scheme to support the notion of strong or weak identities or even anonymity in a totaly decentralized way.

3.1 Routing topology

Directed graphs *(digraphs)* have been extensively used in interconnection networks for parallel and distributed systems design (cf. [23], [24]). Digraphs received special attention from the research community aiming to solve the problem of the so-called (d, k) digraph problem. The goal is to maximize the number of vertices N in a digraph of maximum out-degree d and diameter k [25]. Alternatively, it can be stated as the minimization of the diameter of the digraph given a maximum out-degree and number of vertices N [26]. The well-known Moore-bound applies

[2]Overnet uses a modified version of Kademlia [21], but no complete documentation is currently available.

in every case [27]. An interesting class of digraphs is the lexicographic digraph class, which includes the de Bruijn and Kautz digraphs [13][3].

Before we present the way the de Bruijn graphs are deployed in the presented approach, the topology of those graphs and the way the routing algorithm works is described. Figure 1 shows a $(2,3)$ directed de Bruijn graph denoting a graph with a maximum out-degree of 2, a diameter of 3 and size 8. The maximum number of nodes is always limited by 2^k when the maximum out-degree is two[4]. The graph contains 2^{k+1} directed edges in this case. Each node is represented by k-length (three in this example) strings. Every character of the string can take d different values (two in this example). In the general case each node is represented by a string such as $u_1u_2...u_k$. The connections between the nodes follow a simple left shift operation from node $u_1(u_2...u_k)$ to node $(u_2...u_k)u_x$, where u_x can take one of the possible values of the characters $(0, d-1)$. Figure 1 shows a (2,3) de Bruijn digraph.

The most attractive feature of de Bruijn digraphs (compared to the majority of the DHT-based approaches) is the constant degree requirement for every node. However, this is also their "Achilles' heel" since robustness is difficult to achieve. As an alternative approach in this paper the construction of clusters of peers and the application of the de Bruijn topology for the inter-cluster communication is proposed. This way the nodes of the digraph will be much more stable than single peers. The details of this mechanism are provided in the following sections.

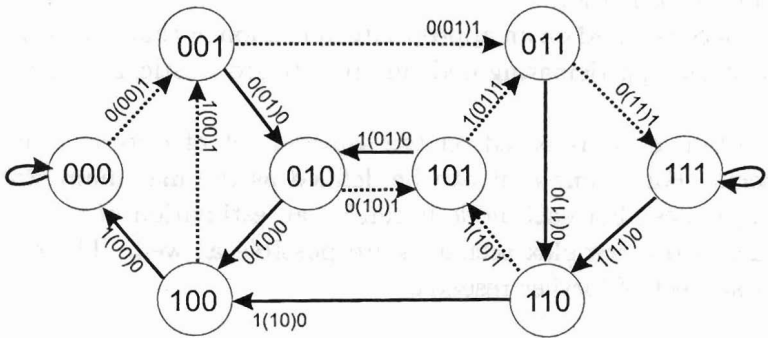

Figure 1: Directed de Bruijn(2,3) graph

3.2 Clustering

Clusters have been introduced into the design of P2P systems in a variety of approaches. JXTA defines the concept of PeerGroups [28] to provide service compatibility and to decompose the large number of peers into more manageable group sizes.

In general clustering algorithms aim mainly at partitioning items into dissimilar groups of similar items. They require the definition of a metric to estimate the similarity of the items in order to perform the partitioning procedure.

For the proposal presented in this paper the clustering algorithm requires criteria other than the usual similarity metrics. The key forces that motivate the construction of the clusters are:

- Clusters that should be composed from nodes (as a group) can fulfill the stability requirements.
- The sizes of the clusters, which should have the smallest possible size in order to reduce the intra-cluster communication complexity.

[3]de Bruijn graphs are less dense than Kautz but they are more flexible since they do not have any limitations on the sequence of the represented symbols in every node.

[4]The Moore bound determines always maximum upper bounds on the size of the graphs that are not reachable for non-trivial cases.

- Clusters are divided when it is possible to create d other stable clusters. Selective division should be applied to maximize the stability of each new cluster.
- Clusters are merged when their estimated stability is lower than a predefined stability threshold.

Of course, a hysteresis step is required to avoid oscillations in splitting and merging clusters.

In order to describe the membership of peers in the clusters a new entity is defined. *Cluster Maps* hold peer identifiers that serve as structures that provide membership information.

Cluster stability estimation: Clusters are defined in a more formal way based on the above rules that govern their existence. Let $\mathcal{T}$ represent the minimum stability threshold required, and $\mathcal{S}_i$ be the estimated stability of the nodes participating in cluster $\mathcal{C}_i$. $\mathcal{S}_i$ is calculated based on s_{ij}, which is the stability of node c_{ij}. Then the definition of $\mathcal{C}_i$ is:

$$\mathcal{C}_i = \{c_{ij} \mid \mathcal{T} \leq \mathcal{S}_i \leq d \cdot \mathcal{T} + \varepsilon\} \quad (1)$$

In Equation (1) d is the maximum degree of the graph and ε is an arbitrarily small value. Note that in the bootstrapping phase the initial cluster is unstable and does not fulfill the conditional inequality of Equation (1).

The calculation of $\mathcal{S}_i$ is based on an appropriate operation $\mathcal{F}$ that gets as input the individual predicted stability of each participating node in order to stochastically calculate the stability of the whole cluster.

The exact definition of $\mathcal{F}$ is based on the selected set of criteria that suits best at the realized P2P systems. For example, it can be defined as the maximum stability encountered in the participating nodes. For each node it can be an estimation of the predicted remaining uptime of the node. More complex scenarios are possible as well. The definition of optimal stability criteria is subject of further research.

3.3 Entity identification scheme

The introduction of clusters in the design of the overlay networks results in the need of making them identifiable entities. Further, peers have to be be identified as well. A dual identification scheme is proposed to satisfy the identification requirements.

Peers are distinguished by a Globally Unique Identifier (GUID) that is created using secure hash functions. Peer GUIDs have constant length. Clusters are distinguished by expandable GUIDs that follow a de Bruijn-like value assignment. The peculiarity of this scheme is that the length of the identifiers is adapted to the size of the graph. Moreover, the length might differ by maximum one for neighbor nodes in order to fulfill the requirement of incremental extendability and even load-balance.

The benefits of this dual identification scheme are multi-fold. Nodes can be uniquely identified and their actions can be traced when this is desirable. Peers are responsible for their actions. Peers can not "force" responsibilities for indexing certain items since the mapping is not depending on the peer GUID but on the cluster GUID. Neighbor selection is not strictly defined. Peers can select their neighbors from neighbor clusters. The selection can be based on proximity, trust or other specific metrics.

3.4 Role identification

One of the major goals of this work is to provide the mechanisms to support harmonic and fair cooperation among heterogeneous peers. The first step towards this direction is the introduction of peer clusters into the architecture that share the cluster responsibilities. A first approach that could handle heterogeneity is to assign weights to each node of the cluster. The assigned

weights represent their relative abilities and direct the requests of the peers of each cluster proportionally to the assigned weight. While this could be helpful to handle the routing effort it would still have high requirements on each peer since all of them should to some extend participate in every core service and operation. Moreover, micro-payment mechanisms would be required to keep track of the contribution and the usage of the system in order to provide incentives for peers to declare their real capabilities.

To achieve support for a fair and harmonic cooperation of the peers with heterogeneous capabilities and behavior, a novel approach that aims to fit the contribution required by each peer to its physical capabilities and user behavior is proposed in this paper.

Analyzing proposed DHT-based approaches (for example Chord, which assumes homogeneous components) it can be observed that they require every peer to participate in every core overlay operation. Peers should participate in the maintenance of the overlay, in the indexing of the advertised items, in the routing of the requests to their logical destinations, etc. Such an approach provides a fair sharing of the effort among the peers and they are as autonomous as possible. However, the fact that in reality peers do not have similar capabilities and that P2P systems are very unstable it is obvious that there are big challenges in the deployment of these systems. On the other hand, analyzing hierarchical systems like eDonkey results in deployable solutions. Nevertheless, they are unfair, partly centralized and dependable ones.

The main motivation behind the approach introduced here is the fact that it is too costly to design the system in a way where each peer participates in every operation. A set of basic operations is identified that can work independently from each other. These are assigned to the peers with the respective roles. The introduction of peer clusters augments this with a very clear and feasible design. The following basic operations/services have been identified. For particular overlays (e.g. for multicasting services) identification of additional roles might be required.

- **Overlay maintenance.** *Maintainers* perform the most demanding operation since peers are required to maintain complete routing tables, indexing information and cluster organization.

- **Indexing.** *Indexers* are required to handle the indexing responsibilities of the whole cluster in a balanced and co-operative way. Information redundancy is an additional requirement in order to shield the system against the single peer (mis-)behavior. Indexers can provide the final reply to queries when they reach the destination cluster.

- **Routing.** Routing is the most simple operation where *Routers* should forward messages to the neighbor clusters that are closer to the final destination. Combined with the low requirements raised by the de Bruijn digraphs it is a role suitable for any peer, even those that have very low capabilities and an unstable behavior. They participate only in the inter-cluster message forwarding service but they do not have the required information to provide the final reply as Indexers have.

- **Caching.** Although caching is not a basic operation (it is rather considered as advanced operation) it is included in the basic scheme because of the low requirements it poses and the fact that it closes the design gap between the Routers and the other more demanding roles. *Cachers* are expected to perform caching of the indexing information for the most popular items in order to reduce the effort of the Indexers. Studies (i.e. [3]) indicate that there is a zipf-law distribution of the queries in popular content-related P2P systems.

It is clear that different requirements are connected with each role (since this was the goal). Table 1 provides a summary of the requirements for each role in an increasing requirements order. Peers are "promoted" to adopt roles with higher requirements as they prove their stability and they fulfill the physical capability needs.

The questions that arises is why a peer should adopt get a Maintainer role that imposes so high requirements in resource contribution and not the role of a Router that obviously requires lower contribution?

Table 1: Requirements of Roles and servicing rules.

Role	Requirements			Servicing rules
	Routing table	Indexing	Stability	
Router	Inter-cluster (de Bruijn)	No indexing		Routers, Cachers, Indexers, Maintainers
Cacher	Inter-cluster & to Routers	Small size cache		Cachers, Indexers, Maintainers
Indexer	Inter-cluster & to Cachers, Routers	Full indexing	✓	Indexers, Maintainers
Maintainer	Inter-cluster, to neighbor cluster Maintainers & complete intra-cluster	Full indexing	✓	Maintainers

In order to provide a balanced distribution of the cluster workload a set of rules should be defined to give incentives to peers taking the roles that fit best to their capabilities and behavior. This set of rules should be both simple and powerful enough in order to be feasible and efficient. The following list provides such a simplified set where the term class is also used to denote a role. Additional operation related rules can be defined.

- Always request the service from the class with the lower requirements when it is available.
- Do not provide a service to a peer belonging to a class of lower requirements unless no other peer of such a class in the cluster can provide it.
- Always provide higher priority in servicing classes of higher requirements.

These rules regulates the load distribution among the identified classes. For example, a Router shall not forward a query to (or it shall not be served by) a Maintainer. This way Maintainers will not be overloaded with routing requests. Similarly, when a request will reach the destination cluster it will be routed to a Cacher initially so that Indexers will not be overloaded. If the hit in the cache is false then an Indexer will be contacted to provide the requested information and the Cache will be updated. Having reduced the workload of Maintainers and Indexers, they mostly perform the demanding operations of maintaining the overlay and keeping the indexes updated.

Distributed Algorithmic Mechanism Design techniques are formally analyzed and discussed in [29]. The solution in this paper is a practical heuristic that can provide acceptable results. Further analysis is required to quantify those results and provide a solution with finer granularity in fairness and efficiency. This analysis should consider the query and join/leave rates.

4 OVERLAY MANAGEMENT

It is clear from the previous section how the proposed mechanisms meet certain requirements. Using de Bruijn digraphs, **scalability** is inherent in the design. Routing table size is independent of the system size. Further, the number of hops required by the routing mechanisms to reach any destination increases logarithmically with respect to the system size. Moreover, the simple left shift operation offers a low complexity and **efficient routing** algorithm. **Network proximity** is achieved by peer clustering. Peers can select their neighbors from the set of peers of the neighbor clusters based on the latency of their connection (considering the rules for the

different roles). Additionally, **heterogeneity** is provided primarily by the role-based clustering scheme. Effective **load-balancing** is also provided by the roles and the related rules. In this section, additional algorithms and mechanisms are described based on those described in the previous section. This is necessary to meet the rest of the identified requirements.

4.1 Extendability

De Bruijn graphs have been mainly proposed for interconnection networks of parallel systems that are much more stable and centrally administered compared to P2P systems. They represent excellent selections for static environments. However, special handling is required to make de Bruijn graphs suitable for dynamic environments. In terms of extendability this means that while "complete" de Bruijn graphs can be defined for exponentially increased network sizes (d^k) it is not trivial ensuring their incremental extendability. In the proposed approach a heuristic method is developed, which takes advantage of the proposed mechanisms to deal efficiently with this problem. It should be noted that nodes in the following discussion represent groups and not single peers.

More specifically, the one and two nodes cases are trivial cases and they are complete de Bruijn graphs (of diameter zero and one, respectively). Figure 2 pictures a (2,2) de Bruijn graph (solid nodes and white links). The extended (2,3) de Bruijn graph (dotted links and grey nodes) can be obtained by transforming every link to a node and adding the appropriate connections among them. The identifiers of the new nodes are defined by the concatenation operation on the source of the directed link with the newly shifted bit that leads to the destination. For example, the link that connects the nodes (00) and (01) takes (1) as transition bit, so the newly transformed node will get an identifier of (001).

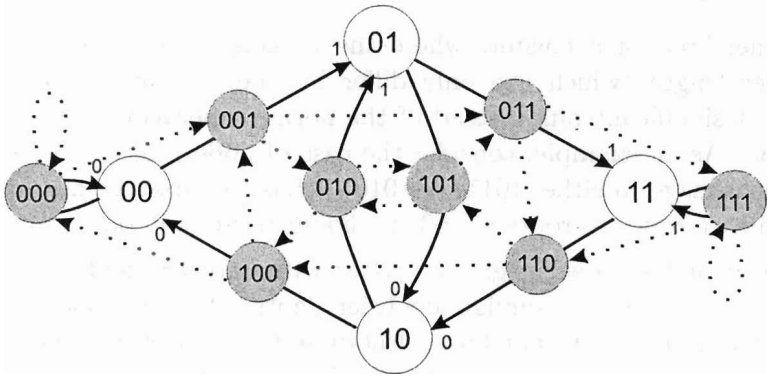

Figure 2: Complete extension of a (2,2) de Bruijn graph to a (2,3) one

The incremental extension of the original graph to the extended one is shown in a detailed illustration in Figure 3. Applying the stability metric on the groups (a local operation performed in every group) some of them can get a high stability value (d times more than the stability threshold). In order to keep the intra-cluster complexity as low as possible, the group is being split in d new groups. The algorithm works as follows:

- Select the node to split:
  - If all of the neighbor clusters have the same identifier depth, then split the initially selected group in d new clusters (by increasing their identifier depth) which are constructed as having the prefix of the initial cluster identifier, concatenated by one additional symbol, which takes one of the d distinct values $(0, d-1)$ (different value for each new group). For example, in Figure 3 (I), node (01) is split into (010) and (011).

- If there is at least one neighbor cluster with shorter identifier depth then this is the cluster that should be split. Redistribute the nodes so that the cluster with the lower identifier will have excessive stability and repeat the previous step on that group[5]. As an example consider the case where cluster (010) fulfills the stability requirements and it can be split. Since clusters (11) and (10) have shorter identifier length, one of them is selected to be split. The case is demonstrated in Figure 3 (II), where cluster (10) is selected to be split into (101) and (100). Further criteria can be applied in the case where more than one neighbor clusters have shorter identifier length.

- Set new connections:
  - If the newly created nodes have longer identifier depth compared to at least one neighbor cluster, then place the same connections as the initial cluster (both for incoming and outgoing links) to all of the d newly created clusters. Incoming connections to the initial cluster are maintained for each new one. This case is shown in Figure 3 (I) where both nodes (011) and (010) have connections with nodes (11) and (10) as (01) did. Additionally, node (10) have connections to both (011) and (010) since initially it had a connection with (01).
  - If the newly created clusters have the same identifier length as all of the neighbors apply the normal de Bruijn connectivity pattern to them and their neighbors. This case is shown in Figure 3 (IV) where the digraph has reached a "complete" state (2,3). It can be easily observed that all nodes have connections similar to the digraph in Figure 1.

- Configure routing mechanism:
  - If the neighbor of a cluster, where the message should be routed to, has longer identifier length (which can only differ by one), make the decision based on the two most significant bits instead of the normal operation that requires a single bit selection. As an example, consider the case of node (10) in Figure 3 (I). In order to route a message to either (011) or (010) it has to parse the next two symbols of the key, while in order to route to (00) it should parse only one symbol.
  - If there is at least one neighbor cluster with shorter identifier length then (if the destination cluster has similar identifier depth) it should make the decision based on the two most significant bits. Otherwise it should select the next neighbor by a single bit operation and ignore its own least significant bit. For example, consider the case of node (100) in Figure 3 (III). In order to route a message to either (011) or (010) it should parse two symbols of the key, while to route the message to node (00), it should parse only one symbol and ignore its least significant bit.

4.2 Overlay stabilization

Removing a cluster is an operation that should be avoided. This is achieved by carefully choosing the right cluster where a new peer shall join. In this phase attention has to be payed to the maintenance of the stability of the clusters, also considering the trade-off between this and the cost for the join operation. As it was suggested above, peers can for example do a random walk in order to select the weakest group to join.

However, there might be cases that clusters are becoming unstable. A special algorithm performs the reverse procedure of creating new clusters in order to minimize the cost of cluster

[5]Actually, this should not be frequently the case if as described above the peer joining process selects groups with lower stability

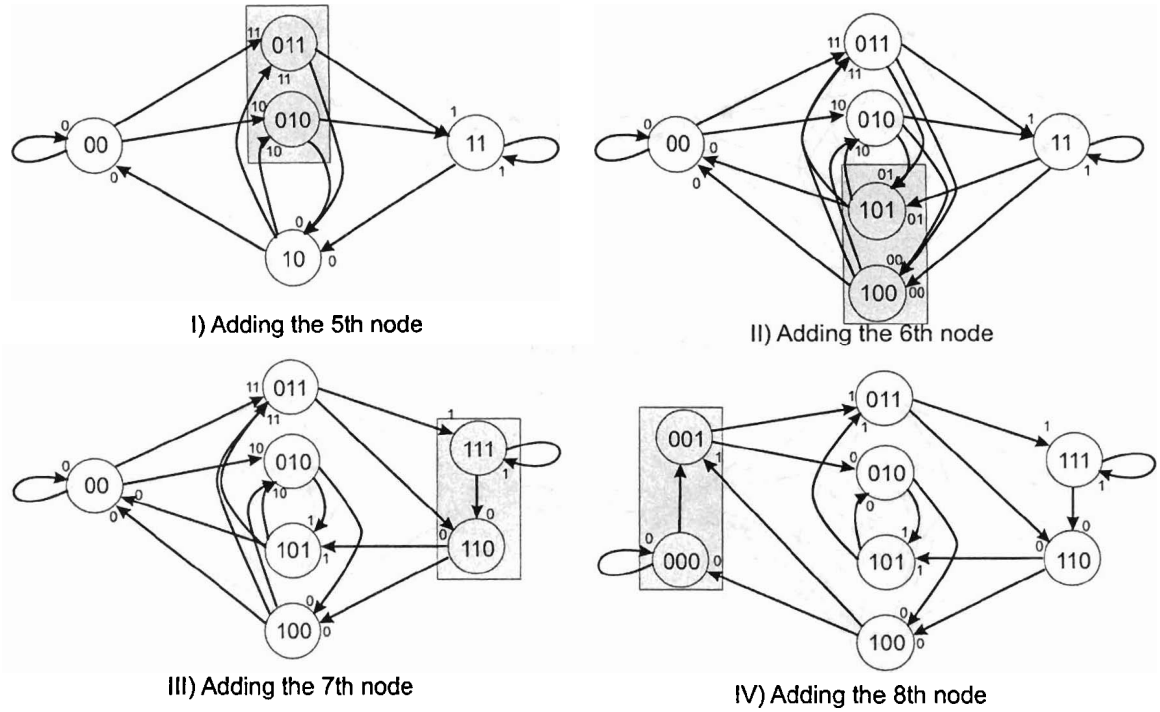

Figure 3: Extending a (2,2) de Bruijn graph, by adding nodes incrementally

"disappearance" when the stability of the cluster is less than the required threshold and no peers can migrate to this unstable cluster. Figure 4 shows two different cases of clusters that should be removed.

- In the first case a cluster that has no neighbors with longer identifier depth is marked as unstable. In this scenario, the d "sibling" clusters that share the same prefix of $(k_{max} - 1)$ length should be merged into one cluster with $(k_{max} - 1)$ identifier length. This is illustrated at Figure 4 (I), where cluster (101) is unstable (left side), so it is merged with (100) to form cluster (10) (right side).
- The second case occurs when it is required to preserve the system invariant that "neighbor cluster identification depth should differ only by one". If the unstable cluster has neighbor clusters with longer identifier depth then one of them is selected from which peers migrate to the unstable cluster. This way the problem is transformed to that of the previous case, since the unstable cluster has the longest identifier depth. This scenario is shown in Figure 4 (II), where instead or removing the initially unstable cluster (11), peers from cluster (100) migrate to (11) and will result in a similar digraph like case (I).

Connections and routing tables are accordingly updated, to reverse the previously described extension procedure.

4.3 Fault-tolerance

Assuming sufficiently effective prediction of node stability, the method introduced in this paper addresses fault-tolerance in a different way than the commonly used high connectivity pattern. Stable clusters are used as components in the de Bruijn overlay that have much smaller join/leave rate than the single-peer based alternatives.

It has been observed at many deployed systems (i.e. [14], [15], [16]) that peers which are connected for a long time tend to stay connected for longer. Taking advantage of these

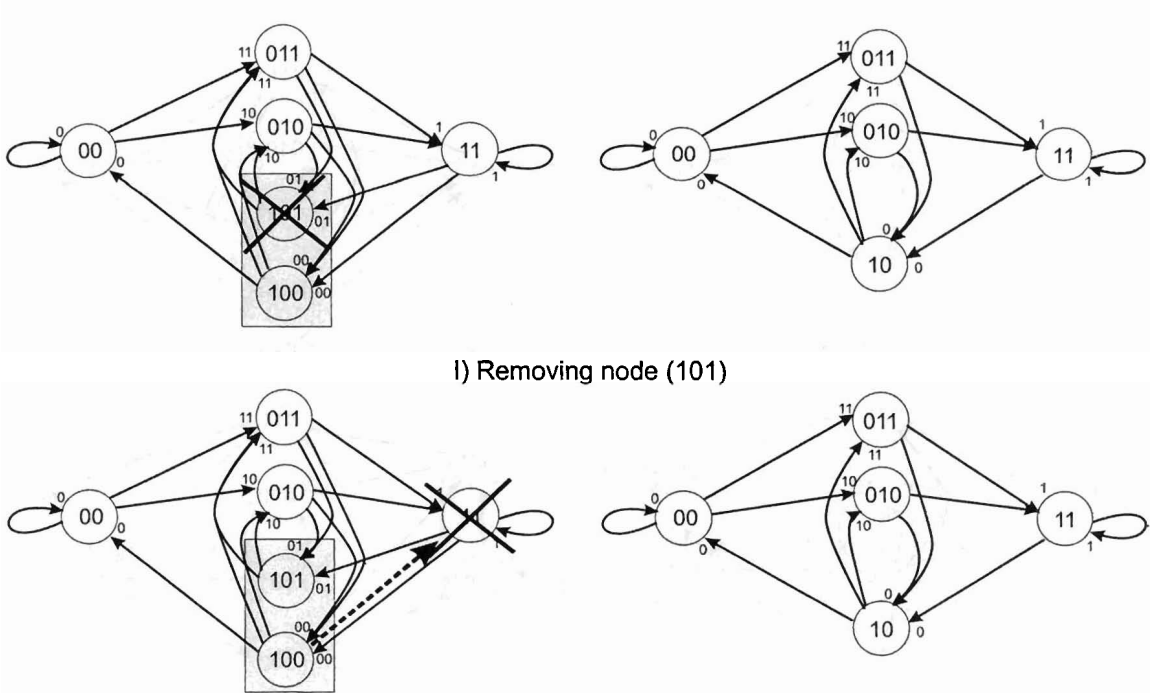

Figure 4: Removing unstable clusters

measurements critical roles are assigned (mainly Maintainers) to predictably more stable peers. This way we relax the commonly introduced requirement within the connectivity pattern where peers should maintain a number of connection that is a logarithmic function of the overlay network size.

The intra-cluster connectivity is higher for the Maintainers but again is limited to the cluster size which is assumed to have reasonable upper bounds. Therefore, the increased connectivity that is applied to obtain fault-tolerance (among other desired properties) is related to the *locally observed peer instability rather than the size of the global system.* Subsequently, the maintenance cost is far smaller than the approaches that employ logarithmically growing connectivity patterns since the update cost increase with the number of neighbors.

Similarly, the structured cooperation of the Indexers and Maintainers ensures the consistency and the availability of the indexing information.

5 OMICRON

Omicron (*Organized Maintenance, Indexing, Caching and Routing for Overlay Networks*) is a P2P system that follows the proposed design guidelines. It is currently being developed to support the communication infrastructure of the MMAPPS project [30]. Additionally, a P2P overlay simulator is being implemented to investigate the behavior of the system under a variety of assumptions and scenarios. In the rest of this section a high-level description of Omicron is provided to illustrate the most important details of realizing the proposed mechanisms.

5.1 Architecture

Following the proposed architecture Omicron is organized in a two-tier, hybrid approach. Peers obtain a unique identifier from a large key space and they are grouped in stable clusters. Clusters

are uniquely identified with identifiers of variable length following the aforementioned procedure. Figure 5 provides a snapshot of a cluster and the connections of its peers to explain Omicron's connectivity patterns both for inter- and intra-cluster communication. Cluster identifiers are related with the left shift operation that correlates neighbor nodes in de Bruijn digraphs (L represents their common part).

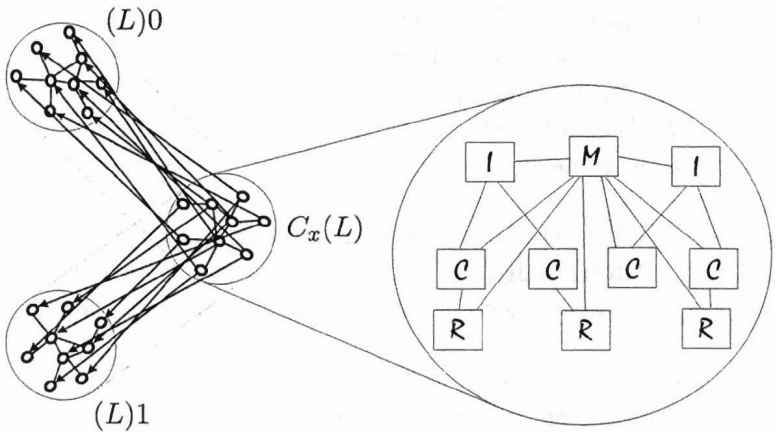

Figure 5: Inter- and intra-cluster connectivity pattern

- **Roles in clusters.** The assignment of the four basic role classes is explained in this example. The Maintainer has a central position. Peers are required to be stable to efficiently fulfill the requirements of this role. Moreover, since the requirements of the Cachers are not much higher (compared to the Routers) it is preferable for efficiency reasons to assign more Cachers than simple Routers.

- **Inter-cluster organization.** In this example peers are required to have a two degree inter-cluster connectivity. Maintainers of neighbor clusters exchange lists of participating peers in each cluster and this information is distributed to the peers. Then each peer selects another one in the neighbor cluster to forward the messages. The mapping can be randomized or follow a certain algorithm to maximize the load-balance.

- **Intra-cluster organization.** In this example not all of the connections among the classes are shown to increase the legibility of the figure. For the same reason connections between peers of the same class have been omitted as well. However, there is a clear need for communication among Maintainers or Indexers to keep the routing tables and the indexing information updated. Here a certain level of redundancy will provide a basic defense against failures.

5.2 Performance Evaluation

In order to evaluate the performance of Omicron a brief analysis is provided in this section. More specifically, the maintenance cost as it is expressed in terms of valid connections that should be maintained and the number of forwarding steps involved in the lookup process will be analyzed.

Every peer should maintain a number of connections inside the cluster it participates and with the neighbor clusters. The number of inter-cluster connections is the same for every peer, independently of the assigned role. When the constructed topology is a complete de Bruijn digraph the number of connections is d. In the case of incomplete digraphs, the number of connections required is d^2 in the worst case.

The number of intra-cluster connections depends on the assigned role. For example, a least demanding role, the Router, should make sure it maintains a valid connection with at least one Maintainer to receive valid updates of the Cluster Map and a Cacher to forward queries that must be replied by the cluster it belongs at. Routers need to have $O(1) + O(d^2)$ number of connections in total (both inter- and intra-cluster). A more demanding role such as Maintainer should maintain a connection with every peer inside the cluster in the worst case. Assuming a pessimistic scenario such as the one in [20] where peers join and leave at a very high rate without taking into account the fact that a small subset remains relatively more stable, the cluster size can asymptotically converge to the logarithm of the network size. For this scenario Maintainers need to have $O(log_d n) + O(d^2)$ number of connections in total. Even for this pessimistic scenario only a small subset of the peers need to maintain a large number of connections to ensure the high robustness of the system.

The lookup operation needs to travel $O(log_d n)$ number of clusters to arrive at the destination cluster in the worst case. Additionally, two more steps might be necessary inside the cluster when the Cacher do not hold the requested item, which should be retrieved from an Indexer of the target cluster.

From this first evaluation it is clear that Omicron is capable of exploring and adapting in the observed stability of the environment to improve performance and reduce the requirements. Even in the worst case scenario, not every peer is required to maintain a logarithmic number of connections with respect to the size of the network.

6 RELATED WORK

There is a number of related proposals with respect to certain aspects of the above described mechanisms.

Koorde [10] for instance is a proposal that deploys the Chord design over de Bruijn digraphs. The authors suggest the construction of de Bruijn graphs with node degree proportional to the logarithmic size of the network to avoid the robustness limitations of constant degree connectivity. This requires a good estimation of the size of the network and it obligates the most attractive feature of the de Bruijn digraphs. Koorde suggests the introduction of "imaginary nodes" to address the incremental extendability limitation of the de Bruijn graphs.

D2B [31] is another Content Addressable Network (CAN) that employs de Bruijn graphs to construct its overlay network. Although the proposed topology is a variation of de Bruijn graphs, they provide an interesting graph operation analysis. In D2B a procedure is suggested that allows nodes to have variable length identifiers of more than one symbol. This is even the case for linked neighbors. The resulting digraph is not always a de Bruijn one.

Rajaraman et al [32] proposed a hierarchical, cluster-based approach which uses de Bruijn graphs to organize the intra-cluster communication scheme. Naor et al [33] proposed de Bruijn graphs for a novel communication architecture.

In order to address the incremental extendability of de Bruijn graphs, Fiol et al proposed a theoretical approach [26] named Partial Line Digraph. Tvrdik [34] described some general algorithms that basically apply for Kautz graphs, but they could be applied in de Bruijn graphs as well. Those techniques could be investigated to improve the topology-related characteristics of our proposal.

Liben-Nowell et al [20] studied and modelled the dynamic behavior of the P2P systems assuming a general probabilistic technique to express the high join/leave rate and made proposals for fault-tolerant overlay design. This approach can not capture effectively the observed measurements on the participation times of the peers. In order to handle the high churn rate, they require a global property in the connectivity pattern that "peers should maintain a number of connections which is a logarithmic function of the overlay network size" [20]. This technique has been analyzed based on the Chord system. Although this approach provides good connec-

tivity guarantees, it poses high requirements on peers capabilities and assumes a homogeneous environment[6].

7 CONCLUSIONS

The paper is addressing the deployment of a number of mechanisms in the design of P2P overlay networks. This is a complex procedure when it targets large scale, highly dynamic and heterogeneous environments. In such scenarios many trade-offs have to be considered between the large variety of distinct requirements. The paper has addressed the most important of those requirements and identified the related trade-offs and the limitations of the current approaches.

We propose a set of mechanisms to provide concurrently *scalability, incremental extendability, fault-tolerance, stability, load-balance, efficiency, fairness and effective collaboration of peers with heterogeneous capabilities and behavior.*

Six major key design mechanisms govern this proposal: (i) An efficient scheme to structure the overlay, based on de Bruijn graphs. (ii) A clustering mechanism that partitions the peers in stable groups that form the components of the overlay. (iii) A role adaptation inside the clusters that effectively matches peers' capabilities and behavior with their responsibilities. (iv) A priority and rule-based service differentiation that provides appropriate incentives for the various roles. (v) A unique combination of structure and randomization in the topology, as it is achieved by the usage of de Bruijn graphs and the selection ability of the the network peer. And finally, (vi) the identification scheme that uniquely identifies each peer without any constraints and groups them under the shared cluster identifier.

Currently Omicron is being developed as a novel communication infrastructure which follows the proposed set of design mechanisms. The behavior of Omicron will be investigated using simulation. In the near future the stability that clusters can achieve will be formally analyzed. Also, the fine tuning of the various parameters that control the presented mechanisms and the extension of the these mechanisms to fulfill further, scenario specific requirements will be investigated based on the simulation.

ACKNOWLEDGEMENTS This work has been performed partially in the framework of the EU IST project MMAPPS Market Management of Peer-to-Peer Services (IST-2001-34201). The authors gratefully acknowledge ongoing discussions with their MMAPPS partners.

References

[1] Gnutella. http://www.gnutella.com, 2003.

[2] eDonkey2000. http://www.edonkey2000.com, 2003.

[3] Nathaniel Leibowitz, Matei Ripeanu, and Adam Wierzbicki. Deconstructing the KaZaa Network. In *3rd IEEE Workshop on Internet Applications (WIAPP'03)*, June 2003.

[4] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM Press, 2001.

[5] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 161–172. ACM Press, 2001.

[6] In addition, the overlay maintenance cost is getting much higher.

[6] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.

[7] B. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz. Brocade: Landmark routing on overlay networks. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, March 2002.

[8] Bernard Traversat, Ahkil Arora, Mohamed Abdelaziz, Mike Duigou, Carl Haywood, Jean-Christophe Hugly, Eric Pouyoul, and Bill Yeager. Project JXTA 2.0 Super-Peer Virtual Network. http://www.jxta.org/project/www/docs/JXTA2.0protocols1.pdf, May 2003.

[9] Jan Mischke and Burkhard Stiller. Rich and Scalable Peer-to-Peer Search with SHARK. In *5th Int'l Workshop on Active Middleware Services (AMS 2003)*, June 2003.

[10] Frans Kaashoek and David R. Karger. Koorde: A simple degree-optimal hash table. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS03)*, February 2003.

[11] N.G. de Bruijn. A combinatorial problem. In *Proceedings of the Koninklije Nederlandse Academie van Wetenshapen*, pages 758–764, 1946.

[12] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 183–192. ACM Press, 2002.

[13] F. Bernabei, V. De Simone, Laura Gratta, and M. Listanti. Shuffle vs. kautz/de bruijn logical topologies for multihop networks: a throughput comparison. In *Proceedings of the International Broadband Communications*, pages 271–282, 1996.

[14] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, 2002.

[15] Balachander Krishnamurthy, Jia Wang, and Yinglian Xie. Early measurements of a cluster-based architecture for p2p systems. In *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement Workshop*, pages 105–109. ACM Press, 2001.

[16] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[17] H. De Meer, K. Tutschku, and P. Tran Gia. Dynamic operation of peer-to-peer overlay networks. *Praxis der Informationsverarbeitung und Kommunication (PIK)*, 2003(2):65–73, 2003.

[18] R. Schollmeier and g. Kunzmann. Gnuviz - mapping the gnutella network to its geographical locations. *Praxis der Informationsverarbeitung und Kommunication (PIK)*, 2003(2):74–79, 2003.

[19] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *ICSI Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[20] David Liben-Nowell, Hari Balakrishnan, and David Karger. Observations on the dynamic evolution of peer-to-peer networks. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, 2002.

[21] P. Maymounkov and D. Maziéres. Kademlia: A Peer-to-peer Information System Based on the XOR metric. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS02)*, 2002.

[22] Overnet. http://www.overnet.com, 2003.

[23] Frank Hsu and David Wei. Efficient Routing and Sorting Schemes for de Bruijn Networks. *IEEE Transactions on Parallel and Distributed Systems*, 8(11):1157–1170, 1997.

[24] Zhen Liu and Ting-Yi Sung. Routing and Transmitting Problems in de Bruijn Networks. *IEEE Transactions on Computers*, 45(9):1056–1062, 1996.

[25] Miguel Fiol, Luis Andres Yebra, and Ignacio Alegre de Miquel. Line Digraph Iterations and the (d,k) Digraph Problem. *IEEE Transactions on Computers*, 33(5):400–403, 1984.

[26] Miguel Fiol and Anna Llado. The Partial Line Digraph Technique in the Design of Large Interconnection Networks. *IEEE Transactions on Computers*, 41(7):848–857, 1992.

[27] W.G. Bridges and S.Toueg. On the impossibility of directed Moore graphs. *Journal of Combinatorial Theory Series B*, 29:339–341, 1980.

[28] Li Gong. Project JXTA: A technology overview, October 2002.

[29] Joan Feigenbaum and Scott Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications*, September 2002.

[30] Market-management of peer-to-peer services. http://www.mmapps.org, 2003.

[31] P. Fraigniaud and P. Gauron. The Content-Addressable Network D2B. Technical Report 1349, LRI, Univ. Paris-Sud, Paris, France, January 2003.

[32] R. Rajaraman, A.W. Richa, B. Voecking, and G. Vuppuluri. A Data Tracking Scheme for General Networks. In *Proceedings of the 13th Annual ACM Symposium on Parallel Algorithms and Architectures*, July 2001.

[33] Moni Naor and Udi Wieder. Novel Architectures for P2P Applications: the Continuous-Discrete Approach. In *Proceedings Fifteenth ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2003)*, June 2003.

[34] Pavel Tvrdik. Necklaces and scalability of Kautz digraphs. In *Sixth IEEE Symposium on Parallel and Distributed Processing*, October 1994.