10. P-Grid: Dynamics of Self-Organizing Processes in Structured Peer-to-Peer Systems

Karl Aberer, Anwitaman Datta, Manfred Hauswirth (EPFL)

10.1 The Concept of Self-Organization

Peer-to-peer systems are often characterized as self-organizing systems. Such characterization is frequently used to informally express properties of Peer-to-Peer systems such as the distribution of control, locality of processing, and the emergence of global structures from local interactions. Self-organizing systems are considered as being particularly scalable and failure resilient.

In this chapter we would like to explore the nature of self-organization in Peer-to-Peer systems in more detail, with a particular emphasis on structured overlay networks. Overlay networks facilitate the organization of application-specific address spaces in Peer-to-Peer systems by constructing a logical network on top of the physical network. They are one of the central concepts that have been introduced in the field of Peer-to-Peer systems. We will investigate the issue of self-organization first for unstructured overlay networks, such as Gnutella [126], where issues of self-organization are more widely studied, and then show how self-organization also plays a role for the design of structured overlay networks. We will study self-organization for the P-Grid structured overlay network [3] which has been designed as a highly self-organizing system.

Self-organizing systems are well-known from many scientific disciplines, in particular from physics and biology, for example, crystallization processes or insect colonies. In computer science self-organization and the resulting phenomena have been studied in particular in the field of artificial intelligence (agent systems, distributed decision making, etc.).

Self-organization is the process of evolution of a complex system with local interaction of system components only, resulting in system states with certain observed or intended global properties. A self-organizing process is driven by randomized local variations—movements of molecules in the case of crystallization, movements of individual insects in insect colonies. These "fluctuations" or "noise", as they are also called, lead to a continuous perturbation of the system and allow the system to explore a global state space until it enters into equilibrium states. These states correspond to the global, emergent structures [294].

R. Steinmetz and K. Wehrle (Eds.): P2P Systems and Applications, LNCS 3485, pp. 137-153, 2005.

More formally, a self-organizing system can be described as a Markovian process. Given a set of possible states S, which usually is very large, the evolution of a complex system can be described deterministically by a function $f_T : S \rightarrow S$. In practice, the lack of information about the precise state will make a deterministic description of the system evolution infeasible. Thus a more realistic way to describe the system evolution is by a stochastic process where for each given state s_i we can give the probability that a state s_j is reached, i.e. $P(s_j|s_i) = M_{ij} \in [0, 1]$, where M is the transition matrix of a Markovian process. Given the probability distribution of states $P(s_i, t)$ at time t it is thus possible to calculate the time evolution of the system as

$$P(s_j, t + 1) = \sum_i M_{ij}\ P(s_i, t).$$

Usually we are interested in emergent properties of such self-organizing systems. Emergent properties are global properties of the state space that result after the system has converged to an equilibrium. In the following we will demonstrate how this view of self-organizing systems can be adopted in the context of Peer-to-Peer systems.

10.2 An Example of Self-Organization in Unstructured Peer-to-Peer Systems

Unstructured Peer-to-Peer systems have generated substantial interest because of their self-organization behavior resulting in interesting global structural properties of their state, in particular the structure of the resulting network graphs. For example, the Gnutella network graph exhibits the following characteristics [516]:

1. The network has a small diameter, which ensures that a message flooding approach for search works with a relatively low time-to-live (approximately 7).
2. The node degrees of the overlay network follow a power-law distribution. Thus few peers have a large number of incoming links, whereas most peers have a very low number of such links.

These properties result from the way Gnutella performs network maintenance: Each peer discovers other peers by constrained flooding. From the discovered peers a fixed number of randomly selected peers are used to construct the Gnutella network. Thus nodes in the network graph have a constant out-degree. During this process peers with a larger number of incoming connections are more likely to be selected. This corresponds to a *preferential attachment* mechanism during network construction. Preferential attachment has been identified as a mechanism that generates a power-law distribution of node degrees for many types of networks, for example, the World Wide

Web, citation networks, and genetic networks. Similarly, for unstructured overlay networks this mechanisms leads to a power-law distribution of nodes' in-degrees.

A non-rigorous argument why preferential attachment generates power-law distributed node degrees is as follows [422]. We model the system state by the distribution of node degrees. Let $P(j,t)$ be the probability that at time t a node has in-degree j and assume that at each time step one peer joins the network and adds one additional connection. Assume that with probability α the node to which the new peer connects is chosen uniformly randomly and with probability $1-\alpha$ proportionally to the current node degree. Then the process of network evolution can be modeled as follows.

$$P(j,t+1) = P(j,t)+\alpha(P(j-1,t)-P(j,t))+(1-\alpha)((j-1)P(j-1,t)-jP(j,t)).$$

Now assume that the degree distribution is in steady state, i.e. $P(j,t) = c_j, t > 0$. We can derive

$$\frac{c_j}{c_{j-1}} = 1 - \frac{2-\alpha}{1+\alpha+j(1-\alpha)} \approx 1 - \frac{2-\alpha}{1-\alpha}\frac{1}{j}$$

where the approximation is valid for large j. This relationships is satisfied approximately for

$$c_j \approx j^{-\frac{2-\alpha}{1-\alpha}}.$$

To see this, note that for this c_j we have

$$\frac{c_j}{c_{j-1}} \approx (1-\frac{1}{j})^{\frac{2-\alpha}{1-\alpha}} \approx 1 - \frac{2-\alpha}{1-\alpha}\frac{1}{j}$$

This is a first example of how a self-organization process results in a global structural feature, namely the power-law degree distribution. The probability that a node has a given in-degree remains invariant while the network grows, thus the system is in a dynamic equilibrium during network construction.

The structure of the resulting overlay network is the basis for performing searches efficiently. In Gnutella, searches are performed by message flooding. A low network diameter, as in the power-law graph, guarantees low search latency. Message flooding however induces a high consumption of network bandwidth. Therefore other strategies for performing searches in Gnutella networks have been investigated. The independence of the network maintenance and search protocols makes it possible to use alternative search strategies which may exploit the emergent overlay network structure more efficiently. Examples of such alternative strategies are the random walker model [397] and the percolation search model [537], which both exploit the specific structure of the network.

To summarize, we can observe two important points for unstructured overlay networks such as Gnutella. First, the structure of the network and

its global properties are induced by the (self-organizing) dynamic process used for their construction. Second, the design of efficient search algorithms exploits the structural features of the overlay network that results from the self-organized construction process. In the following we will show that the same principles can be applied analogously for structured overlay networks.

10.3 Self-Organization in Structured Peer-to-Peer Systems

One of the important drawbacks of unstructured overlay networks is the high network bandwidth consumption during searches, apart from the fact that successful searches are not guaranteed unless all peers are contacted. This motivated the development of structured overlay networks where nodes coordinate among themselves by partitioning the key space and maintaining state information on the resources stored at neighboring nodes. This enables the implementation of directed searches and thus to dramatically reduce bandwidth consumption used in search. This approach, however, requires also a higher degree of coordination among the nodes while constructing and maintaining the overlay network.

To achieve this coordination, maintenance algorithms are provided that maintain structural invariants during the lifetime of the overlay network. Typically these structural invariants are ensured through localized operations. This is the approach taken by most structured overlay networks, such as Chord [575]. The structural invariant of Chord is related to the selection of routing table entries. Each node maintains a link to the first node located on predefined partitions of the key space, which are increasing exponentially in size with distance from the node. During network maintenance, for example, during a node join, the routing tables of the joining node and existing nodes in the network are updated immediately such that after the join is completed, the structural constraints on the routing tables are satisfied. This approach is very different from the self-organization mechanism we have analyzed for unstructured networks, where a structural property, i.e., the node in-degree distribution, is ensured not through localized operations but as a property resulting from a self-organization process.

In the following we will show how self-organizing processes can also be used in the context of structured overlay networks. With such an approach structural properties are not guaranteed through localized operations, but emerge as a global property from a self-organization process. We will demonstrate this approach for the P-Grid overlay network [3]. P-Grid uses self-organizing processes for the initial network construction to achieve load-balancing properties as well as for maintenance to retain structural properties of the overlay network intact during changes in the physical network. We will give first an overview of the structural design of P-Grid and then discuss its self-organization mechanisms and the techniques used for their analysis. Our

focus is on the exemplification of general self-organization principles which are applicable to many overlay networks to varying extents and with different performance implications as we will briefly discuss at the end of this chapter.

10.3.1 The Structure of P-Grid Overlay Networks

We assume that the data keys are taken from the interval $[0, 1[$. The structure of a P-Grid overlay network is based on two simple principal ideas: (1) the key space is recursively bisected such that the resulting partitions carry approximately the same workload. Peers are associated with those partitions. Using a *bisection approach* greatly simplifies decentralized load-balancing by local decision-making. (2) Bisecting the key space induces a canonical trie structure which is used as the basis for implementing a standard, distributed *prefix routing scheme* for efficient search.

This is illustrated in Fig. 10.1. At the bottom we see a possible skewed key distribution in the interval $[0, 1[$. We bisect the interval such that each resulting partition carries (approximately) the same load. Each partition can be uniquely identified by a bit sequence. We associate one or more peers (in the example exactly two) with each of the partitions. We call the bit sequence of a peer's partition the peer's path. The bit sequences induce a trie structure, which is used to implement prefix routing. Each peer maintains references in its routing table that pertain to its path. More specifically, for each position of its path, it maintains one or more references to a peer that has a path with the opposite bit at this position. Thus the trie structure is represented in a distributed fashion by the routing tables of the peers, such that there is no hierarchy in the actual overlay network. This construction is analogous to other prefix routing schemes that have been devised [491, 527]. Search in such overlay networks is performed by resolving a requested key bit by bit. When bits cannot be resolved locally, peers forward the request to a peer known from their routing tables.

P-Grid uses replication in two ways in order to increase the resilience of the overlay network when nodes or network links fail. Multiple references are kept in the routing tables, thus providing alternative access paths, and multiple peers are associated with the same key space partitions (*structural replication*) in order to provide data redundancy. The self-organization mechanisms we will discuss for P-Grid will relate to these two replication mechanisms.

Contrary to standard prefix routing approaches P-Grid does not assume a maximal key length that limits the tree depth and thus search cost. This assumption would compromise the load-balancing properties achieved by bisection. Thus search efficiency is not guaranteed structurally, since in the worst case search cost is related to the maximal path length of the trie, which for skewed key distributions can be up to linear in the network size.

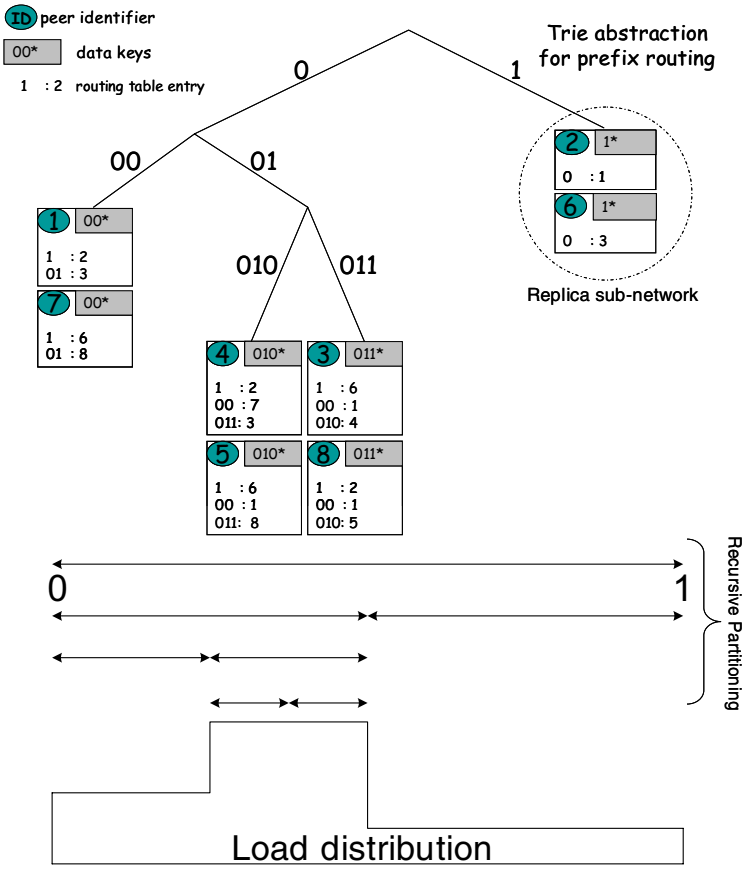

Fig. 10.1: P-Grid structure

To guarantee efficient search, P-Grid constructs its routing tables in a randomized fashion. For example, all peers that adopted 0 as the first bit of their path, can choose at the first level of their routing table any peer with first bit 1. This is in contrast to a deterministic approach where a specific peer would have to be chosen, e.g., the first in the respective interval. It turns out that with such a randomized approach in expectation routing cost is exactly $\log n$, where n is the number of leaves of the trie underlying the routing table construction. Informally, by randomly selecting in the routing tables peers among all peers that can resolve the next bit during routing, in expectation more than one bit will be resolved in a single step. An analysis for the routing cost is found in [2].

There is another motivation for having a trie-structured overlay network instead of a standard distributed hash table: The real advantage of tradition-

ally using a hash table in main memory is the constant time of lookup, insert, and delete operations. But to facilitate this, a hash table sacrifices the order-relationship of the keys. However, over a network, where only parts of the hash table are stored at each location, we need multiple overlay hops anyway. For most conventional DHTs the number of hops is logarithmic in the network size. Thus the main advantage of constant-time access no longer exists. In fact the fundamental issue to address now is, whether we can realize a search tree, which still is similarly efficient as a DHT in terms of fault-tolerance, load-balancing, etc., but also provides properties such as preservation of key ordering and hence supports efficient exact queries but also efficiently enables higher-level search predicates such as substring search, range queries [157], etc. This is a major goal in the design of the P-Grid overlay network.

10.3.2 Dynamics of P-Grid Overlay Networks

From a structural perspective P-Grid is based on a distributed trie structure that preserves key ordering. This naturally enables prefix and range queries [157]. However, a number of algorithmic issues need to be addressed in order to adapt the structure of the overlay to a given key distribution and to maintain it in a dynamic network environment.

Usually maintenance for overlay networks is considered for a *sequential node join and leave model*. Algorithms are provided for inserting new nodes contacting an existing network node. The arrival of a new node requires updates to the routing tables of the new node and of some of the existing nodes in the network. In case of node failures or uncooperative node departure, routing table entries need to be repaired. This is performed by maintenance algorithms which either periodically poll routing table entries or piggy-back repair actions to other operations, e.g., searches.

In addition to these standard techniques P-Grid also supports the efficient construction of an overlay network from scratch. We call this the *bootstrapping problem* and it corresponds to the standard database problem of index construction. For addressing this problem, it is critical that the large number of peers participating in the construction of the overlay network can work *in parallel*.

This problem has been largely ignored in the literature, but has to be solved for a number of practical reasons, in particular for data-oriented applications. The need to bootstrap a new overlay structure can occur for semantic needs of applications, e.g., for indexing new attribute types of resources with structured metadata annotations or due to performance considerations, e.g., for periodically rebuilding an inverted file in a Peer-to-Peer retrieval system rather than continuously maintaining it, or for operational reasons, e.g., for rebuilding an overlay network after catastrophic failure, where the standard

maintenance method is no longer capable of reconstructing a stable overlay network.

In the following we will first discuss a self-organizing process that is used to realize efficient bootstrapping of P-Grid networks, and then discuss a routing table maintenance technique used in P-Grid for maintaining consistency of routing tables under network churn which is based on an adaptive, self-organizing process.

10.3.3 Bootstrapping a P-Grid Overlay Network

The process of bootstrapping an overlay network from scratch should be performed with low latency, i.e., highly parallel, and with minimal bandwidth consumption. At the same time, for a P-Grid overlay network it should simultaneously achieve two load-balancing properties:

1. The partitioning of the search space should be such that each partition holds approximately the same load, e.g., measured as the number of keys present in the partition.
2. Each resulting partition should be associated with approximately the same number of peers, such that the availability of the different data keys is approximately the same.

P-Grid achieves these goals by a distributed, self-organizing process. Using a self-organizing process allows to largely decouple the operation of different peers and thus enables the parallelization of the task.

The design of the process takes advantage of the fact that a P-Grid overlay network structure results from the recursive bisection of the key space. The process is based on random encounters of peers. These are initiated by performing random walks on a pre-existing unstructured overlay network. In their encounters the peers decide whether the current partition contains a sufficient number of keys to justify a further split. The problem to solve is that a large number of peers have to split fast into two peer populations of which the ratio matches the ratio of the key set sizes in the two partitions.

We provide the algorithm used for the basic case of performing a bisection into two partitions 0 and 1 with $n + 1$ peers, where the workload associated with the two partitions is p and $1 - p$. We assume that each peer knows the value of p and that $0 \leq p \leq \frac{1}{2}$. Then the problem can be formulated as follows.

1. *Proportional replication:* Each peer has to decide for one of the two partitions such that (in expectation) a fraction p of the peers decides for 0 and a fraction $1 - p$ for 1. Thus the average workload becomes uniformly distributed among the peers.

2. *Referential integrity:* During the process each of the peers has to encounter at least one peer that decided for the other partition. Thus the peers have the necessary information to construct the routing table.

The second condition makes the problem non-trivial, since otherwise peers could simply select partition 0 with probability p and 1 otherwise. P-Grid uses the following distributed algorithm to solve the problem.

1. Each undecided peer initiates interactions with a uniformly randomly selected peer until it has reached a decision.
2. If the contacted peer is undecided the peers perform a balanced split with probability $0 \leq \alpha(p) \leq 1$ and maintain references to each other.
3. If the contacted peer has already decided for 1 then the contacting peer decides for 0 with probability $0 \leq \beta(p) \leq 1$ and with probability $1-\beta(p)$ for 1. In the first case it maintains a reference to the contacted peer. In the second case it obtains a reference to a peer from the other partition from the contacted peer.

We can model this algorithm as a Markovian process. We assume that in each step i one peer without having found its counterpart so far contacts another randomly selected peer. We denote by $P(0,t)$ and $P(1,t)$ the expected number of peers that have decided in step t for 0 and 1 respectively. Initially $P(0,0) = P(1,0) = 0$. At the end of the process at some step t_e we have $P(0,t_e) + P(1,t_e) = n+1$. We analyze the case $\alpha(p) = 1$. Then the model can be given as

$$\begin{aligned} P(0,t) &= P(0,t-1) + \frac{1}{n}(n - P(0,t-1) - (1-\beta)P(1,t-1)) \\ P(1,t) &= P(1,t-1) + \frac{1}{n}(n - \beta P(1,t-1)) \end{aligned}$$

In order to determine the proper value of β for a given value of p, we have to solve the recursive system. By standard solution methods we obtain

$$\begin{aligned} P(0,t) &= \frac{n}{\beta}(2\beta - 1 + (1-\frac{\beta}{n})^t - 2\beta(\frac{n-1}{n})^t) \\ P(1,t) &= \frac{n}{\beta}(1-(1-\frac{\beta}{n})^t) \end{aligned}$$

We observe that the recursion terminates as soon as no more undecided peers exist, i.e., as soon as $P(0,t_e) + P(1,t_e) = n+1$. By evaluating this termination condition we obtain

$$t_e(n) = \frac{\log(2)}{\log(\frac{n}{n-1})} + 1 \tag{10.1}$$

Note that t_e does not depend on p, and thus the partitioning process requires the same number of interactions among peers independent of load distribution. By definition $p = \frac{P(0,t_e)}{n+1}$, thus we obtain a relationship among the network size $n+1$ and the load distribution p with $\beta(p,n)$. For large networks, by letting $n \to \infty$, we obtain the following relationship among p and $\beta(p)$

$$p = 1 - \frac{1}{\beta}(1 - 2^{-\beta}) \tag{10.2}$$

Positive solutions for $\beta(p)$ cannot be obtained for all values of p. From Equation 10.2 we derive that positive solutions exist for $p \geq 1 - \log(2)$. Informally speaking, since balanced splits are always executed unconditionally, the algorithm cannot adapt to arbitrarily skewed distributions. Therefore for $0 \leq p < 1 - \log(2)$ we have to pursue a different strategy, by reducing the probability of balanced splits, i.e. $\alpha(p) < 1$. The analysis of this case is analogous and therefore we omit it here.

Various non-trivial issues still need to be addressed to extend this basic process to a complete method for constructing a P-Grid overlay network with load-balancing characteristics. The value of p is normally not known, thus it needs to be estimated from the key samples the peers have available locally. This introduces errors into the process which require non-trivial corrections. The process needs to be performed recursively, thus errors in proportionally bisecting the key space accumulate. The process needs to be approximately synchronized to leave the assumptions made for the basic process valid. The bisection process should terminate as soon as the number of peers in the same partition falls below a threshold. Since peers cannot know during the bootstrapping all potential replica peers in the same partition, other criteria, based on the locally available keys, need to be evaluated. Solutions for these problems have been developed and it has been shown that in fact it is possible to efficiently construct a P-Grid overlay network satisfying the desired load-balancing properties based on the elementary process introduced in this section [6, 7].

10.3.4 Routing Table Maintenance

Another aspect of overlay network dynamics is related to the dynamics of the underlying physical networks. Entries in routing tables can turn stale due to temporary or permanent failures of peers or network connections. Standard approaches that address this problem use periodic probing or correct entries immediately upon changes (correction-on-change).

These approaches are specifically designed for environments where peer reliability is relatively high. For P-Grid we assume the contrary, i.e., peers

are generally unavailable. Therefore P-Grid relies on a high degree of redundancy in the routing tables, such that with high probability routing can be performed successfully. Trying to keep all redundant routing entries continuously consistent would not be appropriate in a highly dynamic network. P-Grid rather uses a lazy approach, where routing entries are only corrected if routing fails. We distinguish two possibilities when to perform a repair: (1) immediate repair of stale routing table entries (among a large number of redundant entries) encountered during routing (correction-on-use) and (2) initiate repairs upon failure of all redundant entries at one level of the routing table of a peer (correction-on-failure) [5].

Using a lazy repair approach one can tolerate a certain fraction of stale routing entries in the routing tables. This has the advantage that routing entries to peers that are only temporarily unavailable and reappear, do not require a repair. Furthermore, peers can maintain their path and thus the keys they store also in case of temporary absence, which further reduces the maintenance cost.

In order to enable repairs, P-Grid uses the overlay network itself as a directory for storing the current binding of logical peer identifiers to their current physical address. Peers joining the network have to provide this data. For repair then the binding can be retrieved from the overlay network. Note that during repairs more failures may occur, such that repairs may recursively trigger other repairs. Recursive triggering queries for repair has an inherent self-healing property. With few stale mappings, there is hardly any deterioration in answering the queries, but as the stale entries accumulate over time, they lead to more frequent recursions. An important question is whether such a system can operate in a stable state. For analyzing this, we will model the overlay network as a dynamical, self-organizing system.

To illustrate the route maintenance mechanism we first provide a simple example below. Note that though in the example P-Grid is used as a self-referential directory service for storing identity-to-address mappings of peers, the approach is generally applicable and provides a generic self-contained directory service, such that any information about the participating peers (e.g., trust, history, resource meta-data, etc.) can be stored within the system itself.

Figure 10.2 shows a typical state of a P-Grid network.

Peer P_i is denoted by i inside an oval. Online peers are indicated by shaded ovals, offline peers by unshaded ovals. Peers under the same branch are replicas. For example, P_1 and P_7 are both responsible for paths starting with 000. Without loss of generality we assume that the identity of a peer p (Id_p) has a length of 4 bits. Thus P_7 holds the public key and latest physical address mapping about P_1 (updated by P_1) because P_7 is responsible for the paths 0000 and 0001. The shaded rectangle in the upper-right corner of each peer shows the peer IDs that a peer is responsible for, i.e., whose public key and physical address mapping it manages. Note that there exists

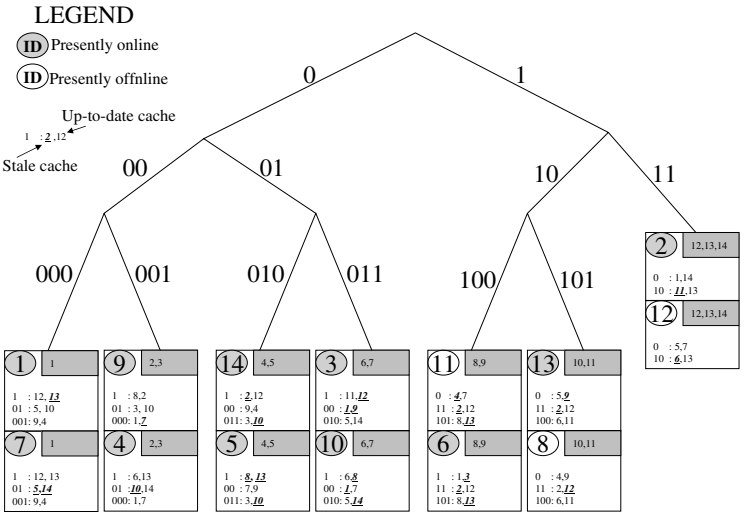

Fig. 10.2: An example P-Grid network

no dependency between the peer identity ($id_{P_7} = 0111$) and the path it is associated with ($\pi(P_7) = 0000$). In its routing table P_7 stores references for paths starting with 1, 01 and 001, so that queries with these prefixes can be forwarded closer to the peers holding the searched information. The cached physical addresses of these references may be up-to-date (for example, P_{13}'s) or be stale (denoted by underlining, for example, $\underline{P_5}$).

A peer P_q decides that it has failed to contact a peer P_s, if one of the following happens: (1) No peer is available at the cached address (trivial case) or (2) the contacted peer has a different identifier. In either of these cases an up-to-date identity-to-address mapping can be obtained by querying the P-Grid. If peer P_s goes offline, and comes online later with a different IP address, it can insert a new identity-to-address mapping into P-Grid.

If a peer fails to contact peers in its routing table, it initiates a new query to discover the latest identity-to-address mapping of any of those peers. If this is successful it forwards the query.

Assuming the initial setup while the P-Grid is in the state shown in Figure 10.2, the query processing will work as follows. Assume that P_7 receives a query $Q(01*)$. P_7 fails to forward the query to either of P_5 or P_{14} since their cache entries are stale. Thus P_7 initiates a recursive query for (P_5), i.e., $Q(0101)$, which needs to be forwarded to either P_5 or P_{14}. This fails again. P_7 then initiates a recursive query for (P_{14}), i.e., $Q(1110)$, which needs to be forwarded to P_{12} and (or) P_{13}. P_{12} is offline, so irrespective of the cache being stale or up-to-date, the query cannot be forwarded to P_{12}. P_{13} is online, and the cached physical address of P_{13} at P_7 is up-to-date, so the query is forwarded to P_{13}. P_{13} needs to forward $Q(P_{14})$ to either P_2 or P_{12}. Forwarding

to P_{12} fails and so does the attempt to forward the query to P_2 because P_{13}'s cache entry for P_2 is stale. Thus P_{13} initiates a recursive query for (P_2), i.e., $Q(0010)$. P_{13} sends $Q(P_2)$ to P_5 which forwards it to P_7 and/or P_9. Let us assume P_9 replies. Thus P_{13} learns P_2's address and updates its cache. P_{13} also starts processing and forwards the parent query (P_{14}) to P_2. P_2 provides P_{14}'s up-to-date address, and P_7 updates its cache.

Having learned P_{14}'s current physical address, P_7 now forwards the original query $Q(01*)$ to P_{14}. This does not only satisfy the original query but P_7 also has the opportunity to learn and update physical addresses P_{14} knows and P_7 needs, for example, P_5's latest physical address (we assume that peers synchronize their routing tables during communication since this does not incur any overhead). In the end, the query $Q(01*)$ is answered successfully and additionally P_7 gets to know the up-to-date physical addresses of P_{14} and possibly of P_5. Furthermore, due to child queries, P_{13} updates its cached address for P_2. Figure 10.3 shows the final state of the P-Grid with several caches updated after the the completion of $Q(01*)$ at P_7.

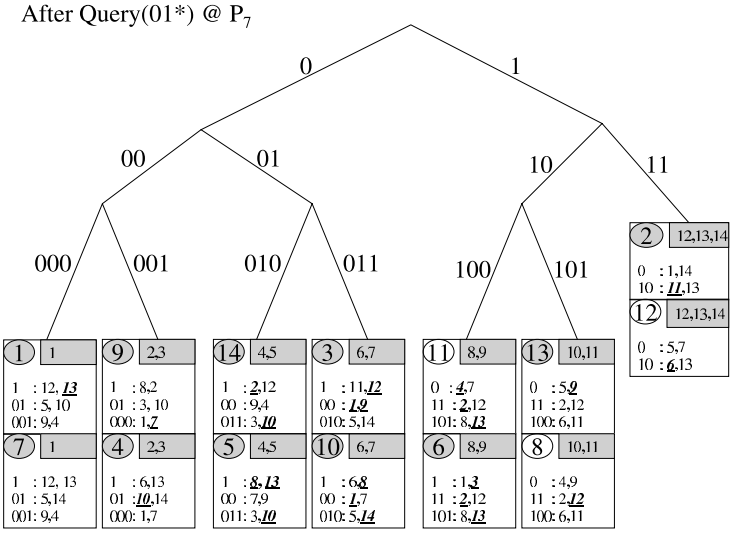

Fig. 10.3: P-Grid after query(01*) at P_7

Peers thus do not discard routing information immediately if it is not usable, since peers may come online at a later time. However, it is also possible that a peer never rejoins the network, and thus a garbage collection mechanism can be used in the background which can be obtained with no or marginal overhead.

10.3.5 Analysis of the Maintenance Mechanism

As the example illustrates, the maintenance mechanism is a highly recursive process. If too many entries become stale the network runs the risk of being no more able to restore correct routing table entries and to catastrophically fail. Thus it is important to understand under which conditions the network remains stable. In order to analyze this we will model the algorithm as a self-organizing process.

We will analyze the dynamics of one of the possible variants of maintenance, eager correction-on-use, in which during routing all routing table entries are probed and get repaired if they are encountered to be stale. We chose this variant here as the analysis is not overly complex. We further simplify the analysis here by considering that only the address changes but the peers always stay online. A complete analysis also considering the probability of peers being online (p_{on}) for both the maintenance mechanisms is much more complicated but uses the same ideas. A detailed analysis taking into account all parameters is given in [5].

We analyze the degree of consistency of routing tables by modeling the time evolution of the probability $P_\mu(t)$ that a entry in a routing table is stale. We assume that at each time step t one query is issued by a peer and a peer changes its address with probability p_c between two queries. The queries issued as a result of the maintenance mechanism will repair a certain fraction of stale routing table entries. The process is in a stable dynamic equilibrium if the expected number of repaired entries matches the expected number of entries becoming stale. The analysis will allow us to determine for which parameters the system is in such an equilibrium state.

In the following n is the number of leaves of the P-Grid tree and r is the number of redundant references kept in a routing table at each depth. We will assume in the following a balanced P-Grid tree.

While cached entries continuously get stale owing to network dynamics, they trigger recursive queries in order to update the stale mappings. In each step of processing a query, an expected number of $rP_\mu(t)$ stale references are encountered and thus trigger a new recursive query. Thus, if we denote by N_{rec} the total number of queries triggered by one original query and consider that in a balanced P-Grid the expected search cost is $\frac{\log_2 n}{2}$ we obtain the recursive relationship $N_{rec} = 1+rP_\mu(t)\frac{\log_2 n}{2}N_{rec}$. The relationship is recursive since each query triggered for repair is expected to recursively trigger more queries.

Not every query (original or recursively triggered) will succeed. Denoting the probability of failure of a query by ϵ, the probability of successfully routing a query is $1-\epsilon = (1-P_\mu(t)^r)^H$ where H is the number of times the query needs to be forwarded to reach the leaf node. Thus, the expected value of the achievable success probability is $1-\epsilon = E_H[(1-P_\mu(t)^r)^H]$. For a balanced

P-Grid, H is a binomial random variable of size $log_2 n$ and parameter 0.5. Hence, $1 - \epsilon \approx (1 - \frac{P_\mu(t)^r}{2})^{\log_2 n}$.

We now can give the Markovian process that determines the time evolution of $P_\mu(t)$.

$$P_\mu(t+1) = P_\mu(t) - p_c(1 - P_\mu(t)) + \frac{1}{r \log_2 n}(N_{rec} - 1)(1 - \epsilon) \qquad (10.3)$$

N_{rec} and ϵ can be expressed in terms of $P_\mu(t)$. The negative contribution in the recursion corresponds to the fraction of correct routing table entries of a peer that turns stale between two queries issued by the peer and the positive contribution is the fraction of incorrect routing table entries of a peer that are repaired due to recursively triggered and successfully processed queries.

The system is in a dynamic equilibrium if $P_\mu(t) = \mu$ for some constant μ.

$$p_c(1 - \mu) r \log_2 n = (N_{rec} - 1)(1 - \epsilon)$$

In this state the rate at which changes occur in the system will equal the rate at which self-maintenance is done due to recursions. This allows us to determine the equilibrium state for different network dynamics expressed by p_c.

In Figure 10.4 we provide contour maps corresponding to N_{rec} values, with p_{on} in the x-axis and $r_{up} = p_c/(1 + p_c)$ in the y-axis. If we consider that there are two kinds of events that trigger the whole self-maintenance process in the system–queries and changes in peers' address mapping–r_{up} represents the fraction of these events being the changes. The interpretation of the plot is thus that if a system is willing to incur an N_{rec} factor of increase of effort per query with respect to the ideal case ($p_{on} = 1$ and $p_c = 0$), the network will operate for all p_{on}, p_c combinations below the curve, with the success probability being 1. If the system is unwilling to use more than N_{rec} effort and if the system operates in the region above the curves of Figure 10.4, there is a non-zero failure probability, which increases with the distance from the curve. Figure 10.4 thus captures two important trade-offs in the system. The first trade-off is that of efficiency versus probabilistic success guarantee of queries. The second trade-off is the system's resilience against the two "demons" of networks, i.e., the network dynamics p_c versus average availability of peers in the network p_{on}.

10.4 Summary

We have seen three examples of how self-organizing processes induce structural features of Peer-to-Peer overlay networks, one example for unstructured overlay networks and two examples for structured overlay networks. Each of

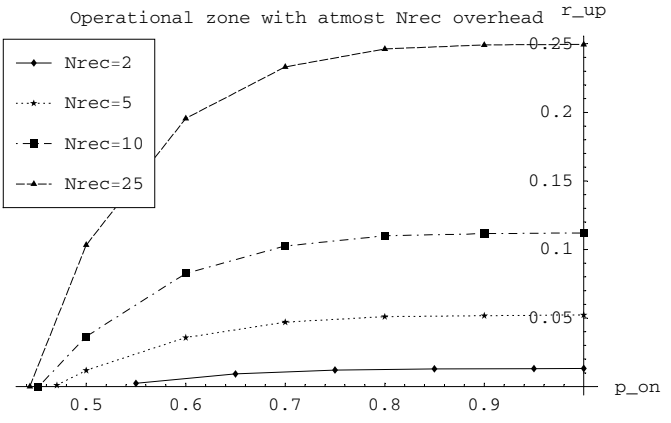

Fig. 10.4: Analytical result: Contour maps for N_{rec}

the examples was slightly different both in the nature of the process studied, in the type of equilibrium obtained as well as in the purpose for which the model of the process was developed.

The ideas presented in this chapter which we explored during the process of designing and implementing the P-Grid system are generally applicable, however. Often dynamic systems will have to be analyzed as Markovian systems, be it for a-posteriori analysis, or to study their evolution over time, given a set of rules for local interactions, or to investigate the equilibrium state in the presence of perturbations.

Also, in the context of Peer-to-Peer systems, reactive route maintenance strategies have been studied by other projects, e.g., in DKS, as well as other systems also try to address the problem of fast and parallel overlay construction mechanisms, for example, [31]. Other systems that focus more on storage load-balancing for arbitrary load distributions and use small-world routing include SkipGraphs [36] and Mercury [72] (among several other recent Peer-to-Peer proposals). Increasingly, there is a confluence of ideas which arrived independently by various research groups dealing with self-organization problems.

More importantly, analysis of network evolution and maintenance are either explicitly or implicitly assuming a Markovian model, an analytical approach which we tried to present here formally by elaborating on three different self-organizing processes. In the case of modeling preferential attachment in unstructured overlay networks the stochastic model has been developed to explain a-posteriori a phenomenon that has been observed in many artificial and natural networks, including Peer-to-Peer overlay networks. Thus it is used to explain empirical evidence. The model itself identifies a dynamic equi-

librium state that is maintained throughout network growth. The dynamics of the network results from the network growth.

In contrast, the stochastic model developed for bootstrapping P-Grid overlay networks is used in order to a priori derive certain design parameters for a distributed algorithm, i.e., the probabilities by which peers take local decisions in order to lead the system globally to a desired state. Also the nature of the stochastic process is different as it is a transient process that converges during network growth to a static equilibrium point. The dynamics of the network results from active interactions performed by peers.

The approach taken for maintenance of P-Grid networks again differs in terms of modeling and methodology. Here the stochastic model is used to determine under which network conditions a stable system behavior for a given algorithm can be expected. As in the first example the stochastic model is used to analyze a dynamic equilibrium state. However, in this case as opposed to the previous examples, the network is stable in size. The dynamics of the network results from operations externally triggered by failures in the physical network.

This illustrates that the basic concept of modeling the large-scale behavior of Peer-to-Peer networks as self-organizing, dynamic processes leaves ample room for different modeling and analysis approaches and has various methodological uses. We would also like to point to the fact that the analysis techniques presented in this chapter mostly have involved substantial approximations and do not necessarily conform to a rigorous mathematical treatment for analyzing dynamical systems. In that respect there exists a substantial need and potential for further interdisciplinary research drawing on the substantial body of knowledge from studying self-organizing systems in other domains. Thus the field of modeling and analyzing complex Peer-to-Peer systems as self-organizing systems currently is still in its early stages.