Modelling Peer-to-Peer Data Networks under Complex System Theory

Cyrus Shahabi and Farnoush Banaei-Kashani
Computer Science Department
University of Southern California
Los Angeles, California 90089-0781
[shahabi,banaeika]@usc.edu

Abstract: A Peer-to-peer Data Network (PDN) is an open and evolving society of peer nodes that assemble into a network to share their data for mutual benefit. PDNs are enabled by distributed query processing. We argue that with a self-organizing, dynamic, and large-scale architecture and nodes that inherit extensive autonomy from their human users, this new generation of distributed database systems should be upgraded from the domain of traditional distributed computing systems to the realm of natural complex systems (such as social networks). In this way, PDN is studied among its peers within a modelling framework which is both compatible with its natural/autonomous computing model and rich to capture its complexity. The "complex system theory" is a meta-theory that provides a common modelling framework to study such complex systems under one umbrella. This meta-theory consists of a rich collection of tools adopted from various fields, where each tool is developed to study an instance of a complex system in a particular field, but applicable across other complex systems from different fields.

In this paper, for the first time we introduce and apply the complex system theory as a modelling framework to PDNs. We demonstrate the usefulness of this modelling framework by presenting a case study, focused on the problem of efficient search in PDNs. Observing the similarity between PDNs and social networks, we adopt a model from the study of social networks to develop an efficient search mechanism for PDNs. More specifically, we propose the SWAM access method, a distributed index structure that enables efficient processing of various similarity search queries (namely, exact-match, range, and kNN queries) in *indexable* PDNs. SWAM is inspired by the "small-world" models originally introduced to explain efficient communication in social networks. We verify the efficiency of these search mechanism both analytically and empirically.

Keywords: Peer-to-Peer Networks, Complex Systems, Small-Worlds, Distributed Search

Reference to this paper should be made as follows: Shahabi, C. and Banaei-Kashani, F. (xxxx) 'Modelling Peer-to-Peer Data Networks under Complex System Theory', *International Journal of Computational Science and Engineering*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Cyrus Shahabi received his Ph.D. degree in Computer Science from the University of Southern California in 1996. He is currently an Associate Professor and the Director of the Information Laboratory (InfoLAB) at the Computer Science Department at University of Southern California. Dr. Shahabi is the recipient of the 2002 National Science Foundation CAREER Award and 2003 Presidential Early Career Awards for Scientists and Engineers (PECASE). His current research interests include Peer-to-Peer Systems, Streaming Architectures, Geospatial Data Integration and Multidimensional Data Analysis.

Farnoush Banaei-Kashani received his M.S. degree in Electrical Engineering - Computer Networks from the University of Southern California in 2001. He is currently a PhD student at the Computer Science Department at University of Southern California, Information Laboratory. His research interests include distributed query processing in dynamic information networks such as peer-to-peer networks, sensor networks, and the Web.

1 INTRODUCTION

A Peer-to-peer Data Network (PDN) is an open and evolving society of peer nodes that assemble into a network to pool and share their data (or more generally, their resources/objects represented by data) for mutual benefit. By an interesting analogy to a democratic human society, when nodes join the PDN society, while they agree to follow a restricted set of common rules in interaction with their peers (i.e., the social rules governing the PDN society), they preserve their autonomy as individuals. For example, as part of their social obligations all PDN nodes (or at least those who are good PDN citizens) create and maintain connection with a set of neighbor nodes and participate in cooperative query processing (e.g., forwarding search queries for data discovery). Aside from the social rules, the PDN leaves the behavior of the individual nodes unregulated and flexible, to be managed by their users based on their individual preferences and/or to allow for natural uncertainties and constraints. For instance, nodes may join and leave the PDN society as they decide (by user decision or due to unwanted node/link failure), they control their own resources/data, and they select their neighbors according to their own administrative policy or physical constraints (e.g., connecting to the nodes that are both accessible and physically close as neighbors). In this sense, individual nodes are self-governed, autonomous, and independent. There is a trade-off between the extent of the social rules and the autonomy of the individual PDN nodes; the more extensive and interfering the social rules, the autonomy of the nodes is more restricted.

1.1 Motivation: Why Complex Systems?

PDNs are distributed query processing systems representing a new generation of distributed database systems with an open architecture and significantly less constraining assumptions. The first step toward realizing this generation of distributed databases is to select an appropriate approach to model these systems. As a direct consequence of the computing model described above, a PDN is 1) a self-organizing system, i.e., there is no central entity to organize the PDN and any kind of structural and functional organization emerges from the distributed interaction among PDN nodes; 2) a dynamic system, i.e., the node-set, data-set, and link-set of the PDN are dynamic and in continuous renewal; and 3) a large-scale system, because as an open and beneficial society it tends to attract numerous nodes that intermittently join the society. The combination of these three characteristics makes PDN a "complex system", i.e., a system that is hard to represent/describe information theoretically (considering the large amount of information required to represent the state of the system), and hard to analyze computation theoretically (considering the complexity of computing the state transition of the system). An appropriate modelling approach for such complex PDNs must 1) be compatible with the PDN computing model as a democratic society, and 2) provide a framework with a set of conceptual, experimental, and analytical tools to contemplate, measure, and analyze PDNs; a framework which is neither oversimplified nor overcomplicated to remain both accurate and applicable to such complex systems. Therefore, we propose the "complex system theory" as the modelling framework for PDNs.

The complex system theory is a unifying meta-theory for collective study of the *complex* systems. Various fields of study, such as sociology, physics, biology, chemistry, etc., were established to study different types of initially simple systems and gradually matured to analyze and describe instances of incrementally more complex systems. The complex system theory is an interdisciplinary field of study which is recently founded based on the observation that analytical and experimental concepts, tools, techniques, and models developed to study an instance of complex system in one field can be adopted, often almost unchanged, to study other complex systems in other fields of study (Bar-Yam, 1997). This meta-theory provides a common modelling framework consisting of a rich set of tools adopted from various fields to study all complex systems under one umbrella.

In this framework, complex systems are modelled as large-scale networks of functionally similar (or peer) nodes, where the links represent some kind of system-specific node-to-node interaction. For example, a social network is a network of people who communicate in a society, a biological network (at the cellular scale) is a network of cells which exchange mass and energy in a biological organ, and a molecular network is a network of molecules that interact by exchanging kinetic and potential energy. Most of the complex systems studied under the complex system theory are *natural* systems, where nodes are autonomous while they also follow certain natural principles/laws (e.g., the second law of Newton governs kinetic interactions among molecules in a molecular network). Moreover, most of the natural complex systems are also self-organizing, dynamic, and large-scale. All the features discussed above are similar to those of PDN and the PDN computing model. Thus, the perception of the complex system theory about complex systems is compatible with PDNs. With the PDN-compatible system model on one hand, and the rich set of special tools to study complex systems on the other hand, the complex system theory is a promising modelling framework for PDNs.

Previously, this modelling approach is successfully applied to the Internet. For example, Ohira et al. (1998) used self-organized criticality (i.e., a self-similarity model from the complex system theory (Sornette, 2000)) to explain the self-similar scaling behavior of the Internet traffic flows, and Albert et al. (2000) employed concepts from statistical mechanics (which was originally developed by physicists to study the collective behavior of the molecular networks, such as temperature and pressure of a mass of gas) to understand the reasons for the power-law connectivity in the Internet topology. To the best of our knowledge, our work is the first attempt to apply the complex system theory as a modelling framework to PDNs.

1.2 State-of-the-Art

Currently, distributed computing is the framework adopted to model PDNs. With this modelling approach, in line with the traditional system-engineering routine, the system designer implicitly assumes almost full control over the system components and resources. This assumption allows reducing the complexity of the system by imposing fabricated restrictions, and consequently, enables designing efficient mechanisms and architectures. Such an assumption may be valid with typical engineered systems that are managed by a unique authority that governs the entire system. However, it is totally incompatible with the democratic PDN computing model, where autonomy of the nodes is an essential requirement. Hence, with this modelling approach the resulting solutions are unrealistic and inapplicable for the real PDN applications. Such theoretical solutions that enforce the controlling assumption give rise to *dictatorial* PDN societies, which are unattractive for prospective citizens, and intolerant and/or fragile to disobedience of their members that want to maintain their autonomy.

The main representative of such solutions is a family of lookup systems, the Distributed Hash Tables (DHTs) (Ratnasamy et al., 2001, Stoica et al., 2001 and Rowstron et al., 2001), which are designed for efficient search in PDNs. DHTs regulate both the data placement and the network topology of the PDN. With the regulated data placement, it is as if the entire data-set of the PDN is owned by a single authority that collects the data from the nodes (the actual owners) and re-distributes the data among them (as a set of slave data storage units/nodes) according to a certain data placement policy to achieve efficient access. Enforcing the data placement violates the autonomy of the PDN nodes in controlling their own data, and for example, is inapplicable to the PDN applications where nodes must maintain their own and only their own data because of security concerns. Moreover, such an unnatural data distribution (which requires modification to the natural data distribution of realistic PDN applications where each node maintains its own data) is an instance of over-engineered design and raises significant practical issues. For example, the communication overhead of transferring the data (or pointers to the data) from the actual owner of the data to where the data is placed can be overwhelming. This important cost factor, which is due whenever the node joins the PDN or its data is updated, is often overlooked in the analysis of the efficiency of the DHTs.

Similarly, with the regulated network topology, among all possible choices of neighborhood, each node is required to connect to a particular pre-defined set of nodes as neighbors. Enforcing the neighborhood of a node violates the autonomy of the node in selecting its neighbors according to its own administrative policy or physical constraints. For example, it is quite possible that none of the designated neighbors for a node are physically accessible to the node when it joins the PDN; hence, leaving the node isolated. Considering such problems with DHTs, it is not surprising that despite significant efforts of the research community in enhancing and promoting DHTs as the only academic solution for efficient search in PDNs, DHTs are not adopted as practical solutions for any real PDN applications such as file-sharing systems. Instead, these systems have unstructured network topology and prefer to use naive search mechanisms such as flooding, which is not efficient but compatible with the PDN computing model, and hence, practical.

1.3 Contributions

We categorize PDNs as instances of complex systems and apply the complex system theory as a modelling framework to study PDNs. Our general research agenda is to extend application of the complex system theory to PDNs by:

1. Adopting models and techniques from a number of impressively similar complex systems (e.g., social networks) to design and analyze PDNs; and

2. Exporting the findings from the study of PDNs (which are *engineered* complex systems, hence, more controllable) to other complex system studies.

We demonstrate the usefulness of this modelling framework by pursuing two case studies, both focused on the problem of efficient search in PDNs. Observing the similarity between PDNs and social networks, we adopt two models from the study of social networks to develop efficient search mechanisms for two types of PDNs. Search is a generic primitive for query processing in PDNs: a mechanism that locates the required data in response to one or more types of queries is a search mechanism. Developing efficient search mechanisms for the self-organizing, dynamic, and large-scale PDNs is a challenging task. We recognize two different types of PDNs that require significantly different search approaches: unindexable PDNs and indexable PDNs.

With unindexable PDNs, the extreme dynamism of the PDN node-set, data-set and link-set renders any attempt to self-organize the network to an index-like structure (for efficient query processing) impossible and/or inefficient. Without indexing, efficient search is only possible by efficient scanning of the network nodes. For unindexable PDNs, we introduce the *SIR* search mechanism that enables efficient processing of partial selection queries (i.e., selection queries that can be satisfied by a partial result-set rather than the entire result-set). SIR is inspired by the SIR (Susceptible-Infected-Removed) epidemic disease propagation model for social networks. We also employ the percolation theory to formalize and analyze this search mechanism.

On the other hand, with the indexable PDNs, the dynamism of the PDN is such that the benefit of indexing the PDN still exceeds the overhead of maintaining/updating the index. For indexable PDNs, we propose a self-organizing mechanism that structures the PDN to the *SWAM* access-method, a search-efficient structure that enables efficient processing of various similarity queries (namely, exact-match, range, and kNN queries). SWAM is a distributed index structure that organizes the PDN nodes in order to index the data content of the nodes while it avoids changing the natural placement of the data. For the design of SWAM as well as its search dynamics, we were inspired by the "small-world" models. Small-worlds are models proposed to explain efficient communication in social networks. For the remainder of this paper, we focus on our second case study with indexable PDNs.

After a short overview in Section 2, in Section 3 we formally define the problem of similarity-search in PDNs. Section 4 elaborately describes the SWAM family of PDN access methods, and specifies SWAM-V as a particular member of the SWAM family. Section 5 concludes the paper and discusses the future directions of this research.

2 OVERVIEW

In this case study, we formalize the problem of *similarity-search* in indexable PDNs, and propose a *family* of distributed access methods, termed *Small-World Access Methods (SWAM)*, for efficient execution of various similarity-search queries, namely exact-match, range, and k-nearest-neighbor queries. Unlike LH* and DHTs, SWAM does not control the assignment of data objects to PDN nodes; each node autonomously stores its own data. Besides, SWAM supports all similarity-search queries on multiple attributes. SWAM guarantees that the query object will be found (if it exists in the network) in average time logarithmically proportional to the network size. Moreover, once the query object is found, all the similar objects would be in its proximate network neighborhood and hence enabling efficient range and k-nearest-neighbor queries.

As a specific instance of SWAM, we propose *SWAM-V*, a Voronoi-based SWAM that indexes PDNs with multi-attribute data objects. For a PDN with N nodes SWAM-V has query time, communication cost, and computation cost of $O(\log N)$ for exact-match queries, and $O(\log N + \mathbf{s}N)$ and $O(\log N + \mathbf{k})$ for range queries (with selectivity $\mathbf{s}$) and $\mathbf{k}$NN queries, respectively. Our experiments show that SWAM-V consistently outperforms a similarity-search enabled version of CAN in query time and communication cost by a factor of 2 to 3. Here, due to lack of space we omit the details of our analytical and experimental results.

3 FORMAL DEFINITION OF THE PROBLEM

3.1 Data and Query Model

We assume a relational data model for the content of the indexable PDNs. A set of (maybe duplicate) tuples with the same schema are distributed among the nodes of the

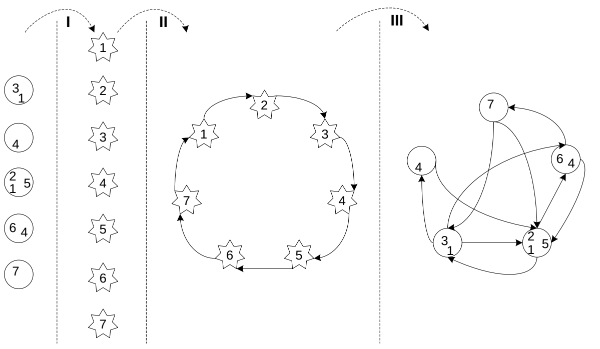


Figure 1: Reducing the general PDN model

PDN (for multi-schema PDNs, we rely on schema reconciliation techniques such as that of Doan et al. (2001)). Tuples are uniquely identified by a set of d attributes, the key of the schema. Hereafter, we use the terms tuple and key interchangeably wherever the meaning is clear. A similarity query is originated at a PDN node and is answered by locating at least one replica of all the tuple(s) with key *similar* to the query key. A PDN access method is a mechanism that defines 1) how to organize the PDN topology (interconnection) to an index-like structure, and 2) how to use the index structure to process the similarity queries. We are interested in the access methods for efficient processing of similarity queries in indexable PDNs.

We model the PDN key space as a Hilbert space $(V, \mathcal{L}_p)$. $V = V_1 \times V_2 \times ... \times V_d$ is a d-dimensional vector space, where V_i, the domain of the attribute a_i for the key $\overrightarrow{k} = \langle a_1, a_2, ..., a_d \rangle$ in V, is a contiguous and finite interval of $\mathbb{R}$. The $\mathcal{L}_p$ norm with $p \in \mathbb{Z}^+$ is the distance function to measure the dissimilarity (or equivalently *similarity*) between two keys $\overrightarrow{k_1}$ and $\overrightarrow{k_2}$ as $\mathcal{L}_p(\overrightarrow{k_1} - \overrightarrow{k_2})$, where $\mathcal{L}_p(\overrightarrow{x}) = \left(\sum_{i=1}^{d} |x_i|^p\right)^{\frac{1}{p}}$.

We are interested in content-based access methods, i.e., access methods that organize the PDN topology based on the content of the PDN nodes. In general each PDN node may include more than one tuple. For better explanation of our content-based access methods, without loss of generality, we find it simple to assume a PDN model where each node stores *one and only one* tuple. To justify this assumption, here we show how to reduce the general PDN model to our assumed PDN model. Consider K as the set of keys (tuples) available in PDN and N as the set of PDN nodes. Assuming a general PDN model, we define a one-to-many mapping $\mathcal{M} : N \to K$ that maps each PDN node to the set of keys stored at the node[1] (Figure 1, Step I). Each key is considered as a *virtual* node embedded in V. Note that since tuples are replicated, there might be several virtual nodes with the same key. A content-based

[1]Depending on the PDN application, if some of the data objects within a node are closely similar, then alternatively $\mathcal{M}$ can map a node to the centroid of the similar objects. Without loss of generality, we focus on the general case where the objects within a node are not closely similar.

access method defines how to organize the set of virtual nodes corresponding to all nodes in N to a *virtual* PDN with particular topology and how to process the queries in the virtual PDN (Figure 1, Step II). Finally, the topology of the actual PDN is deduced by inverse mapping from the topology of the virtual PDN: a PDN node n is connected to a node m if and only if at the virtual PDN some virtual node in $\mathcal{M}(n)$ is connected to some other virtual node in $\mathcal{M}(m)$ (Figure 1, Step III). Also, the semantic of the query processing at the actual PDN nodes is defined by the query processing semantic at the corresponding virtual nodes such that the flow of the query at the actual PDN is logically identical to that of the virtual PDN. With this approach, the mapping and inverse mapping steps (Steps I and III) are independent of the access method used in Step II, and each access method for virtual PDNs (which is a PDN with only one tuple per node) defines an access method with similar characteristics for general PDNs. Hereafter, we assume the reduced model for PDNs and characterize the primitives of an access method to construct the topology/index and process the queries in such a PDN.

The topology of a PDN can be modelled as a directed graph $G(N, E)$, where the edge $e(n, m) \in E$ represents an asymmetric neighborhood relationship in which node m is a neighbor of node n. Schematically, we depict this relationship by drawing an arrow from node n to node m. $\mathcal{A}(n)$ is the set of neighbors for the node n. To achieve scalability, a node only maintains a limited amount of information about its neighbors, which includes the key of the tuples maintained at the neighbors and the physical addresses of the neighbors. A node can directly communicate with its neighbors. To construct the PDN index, an access method defines the *join* primitive[2] (similar to the *insert* operation with the traditional database access methods), which is used by the new node n to delineate $\mathcal{A}(n)$ as it joins the existing PDN. We assume that at least the physical address of one node in the existing PDN (if any) is available to n as it joins the PDN. As the new nodes join the PDN, its topology incrementally converges to the intended index structure. Similarly an access method defines the *leave* operation (equivalent to the delete primitive with the traditional access methods).

We are interested in the following types of similarity queries:

- **Exact-Match Query**: Given the query key $\overrightarrow{q}$, return the tuple t with key $\overrightarrow{k}$ such that $\overrightarrow{k} = \overrightarrow{q}$.
- **Range Query**: Given the query key $\overrightarrow{q}$ and the range $\mathbf{r}$, return all tuples t with key $\overrightarrow{k}$ such that $\mathcal{L}_p(\overrightarrow{k} - \overrightarrow{q}) \leq \mathbf{r}$.
- **k-Nearest-Neighbor (kNN) Query**: Given the query key $\overrightarrow{q}$ and the number $\mathbf{k}$, return the $\mathbf{k}$-ary $(t_1, t_2, ..., t_{\mathbf{k}})$ such that $\overrightarrow{k_i}$, key of t_i, is the i-th nearest neighbor of the key $\overrightarrow{q}$.

[2]This *join* is different from the *join* operation in the relational algebra.

A similarity query can originate from any PDN node at T_0-th time slot ($\forall\, T_0 \in \mathbb{Z}$), assuming a discrete wall-clock time with fixed time unit. A node that originates a query or receives the query from other nodes at the $(T_0 + i)$-th time slot ($\forall\, i \in \mathbb{Z}^+ \cup \{0\}$), can process the query locally and/or forward zero or more processed replicas of the query to its immediate neighbors at the $(T_0 + i + 1)$-th time slot. The collective processing of the query by the PDN nodes is completed when all expected tuples in the relevant result set of the query are visited by at least one of the replicas of the query. Besides the join and leave primitives, an access method defines the *forward* primitive for query processing based on the constructed PDN index. The forward primitive can only use the information at the local node to process the query and to make forwarding decisions. During query processing, the $\mathcal{L}_p$ distance between the query key $\overrightarrow{q}$ and the local key is computed to verify if the local tuple satisfies the query condition. Also, with content-based access methods the forward primitive may measure the $\mathcal{L}_p$ distances between the query key $\overrightarrow{q}$ and the neighbor keys to guide the query.

3.2 Efficiency Measures for PDN Access Methods

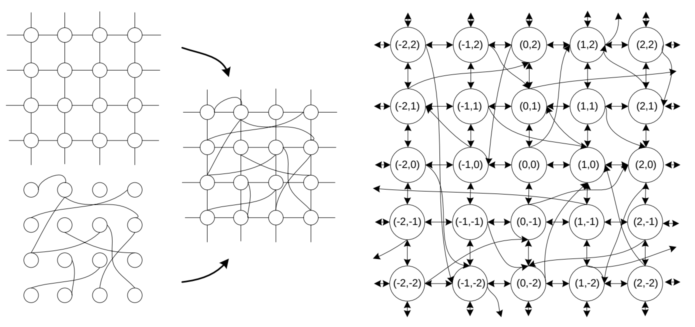
An access method can be evaluated based on its construction cost, and/or based on its query processing cost and performance. Unless the set of nodes participating in PDN is extremely dynamic, the computation (CPU time) and communication costs of constructing and maintaining the index structure are negligible as compared to those of the query processing.

We define three metrics to measure the efficiency of a PDN access method for query processing. The first two metrics evaluate the cost of query processing in terms of the required system resources, whereas the last one measures the system performance from the user perspective:

1. *Communication cost* ($\mathbf{C}_1$): Average number of query replicas forwarded to complete the processing of a query.
2. *Computation cost* ($\mathbf{C}_2$): Average number of $\mathcal{L}_p$ distance computations to complete the processing of a query.
3. *Query time* ($\mathbf{T}$): Average response-time of a query. If processing of a query starts at time slot T_0 and completes at time slot T_1, the response-time of the query is equal to $T_1 - T_0$.

4 SWAM: SMALL-WORLD ACCESS METHODS

We define a family of efficient access methods for PDNs, termed *Small-World Access Methods (SWAM)*, which is designed based on the principles borrowed from the small-world models. Here, after a general overview of the



a. Hybrid small-world graph b. Small-world as PDN index

Figure 2: The small-world model

useful properties of the small-world model, we define the SWAM family and characterize its properties. Also, as an example we introduce SWAM-V, a Voronoi-based instance of SWAM, which satisfies SWAM properties and achieves query time, communication cost, and computation cost logarithmic to the size of the network for all types of similarity queries.

4.1 Small-World as an Index Structure

The small-world model is a network topology proposed to explain the small-world phenomenon, the fact that two individuals in a social network can efficiently locate each other through a short chain of acquaintances logarithmic to the size of the network (Watts et al., 1998 and Kleinberg, 2000). The small-world graph is a hybrid graph, a superimposition of a regular grid and a dilute random graph ($p \ll 1$), inheriting both their properties (see Figure 2-a). It inherits average node-to-node path length $O(\log|N|)$ from the random graph component, and high *clustering* property from the grid. A graph is clustered if the neighbors of a node are more probably the neighbors of each other rather than the neighbors of the other nodes in the network. For a node n clustering is measured by the clustering coefficient $C(n)$, which is the realized fraction of all possible edges among the neighbors of n:

$$C(n) = l \Bigg/ \left(\begin{array}{c} |\mathcal{A}(n)| \\ 2 \end{array} \right) \tag{1}$$

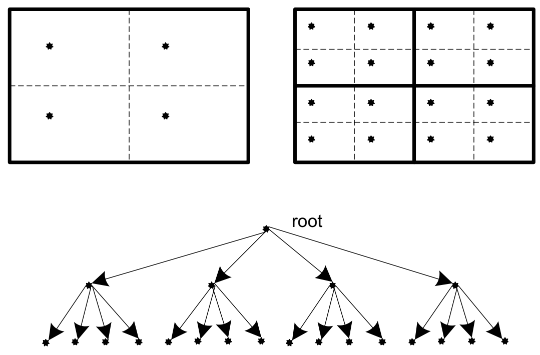
where l is the number of existing edges among the neighbors of n. The clustering coefficient of a graph is the average of the clustering coefficients of its nodes. For a complete graph, a grid, and a dilute random graph $G_{N,p}$, the clustering coefficients are 1, $\simeq \frac{3}{4}$, and $p \ll 1$, respectively.

To demonstrate a direct application of the small-world graph as an index structure for a PDN, we consider the following simple PDN. Assume the key space V is a subspace of $\mathbb{Z}^d$ rather than $\mathbb{R}^d$, and also assume all possible keys in V are available within the PDN, one key per PDN node. We can organize the topology of this PDN based on a small-world graph with a d-dimensional underlying grid as follows:

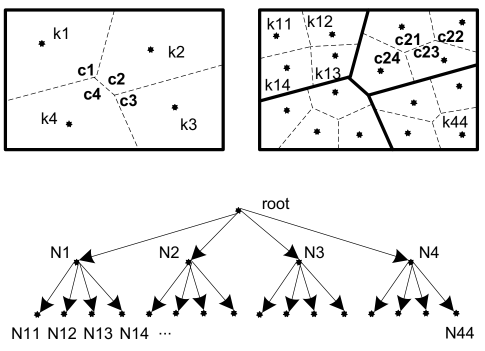
1. Grid component: The node storing the key $\overrightarrow{k} = \langle a_1, a_2 \ , ..., a_d \rangle$ is a neighbor of all nodes with keys $\overrightarrow{k'}$ where $\mathcal{L}_p(\overrightarrow{k} - \overrightarrow{k'}) \leq b$ ($b \in \mathbb{Z}^+$); and
2. Random graph component: The node n_k storing the key $\overrightarrow{k} = \langle a_1, a_2, ..., a_d \rangle$ is a neighbor of one other node $n_{k'}$ with key $\overrightarrow{k'}$ selected probabilistically such that if $\mathcal{L}_p(\overrightarrow{k} - \overrightarrow{k'}) = x$, the probability of selecting n'_k as the neighbor of n_k is proportional to x^{-d} (i.e., a power-law distribution).

See Figure 2-b for an example with 2-dimensional key space, $\mathcal{L}_1$ as the distance measure, and neighborhood boundary parameter $b = 1$. Kleinberg (2000) showed that with a greedy forwarding primitive, on average an exact-match query is resolved with $\mathbf{T}$, $\mathbf{C}_1$, and $\mathbf{C}_2$ all in $O(\log|N|)$. With the greedy forwarding, node n forwards a query $\overrightarrow{q}$ only to one of its neighbors with key $\overrightarrow{k}$ such that $\mathcal{L}_p(\overrightarrow{k} - \overrightarrow{q})$ is minimum among all neighbors in $\mathcal{A}(n)$, i.e., the neighbor with the most similar key to the query key $\overrightarrow{q}$ is selected to receive the query. It is easy to see the underlying grid topology ensures that when a node with key $\overrightarrow{k}$ receives a query $\overrightarrow{q}$, always either $\overrightarrow{k} = \overrightarrow{q}$ or the node has at least one neighbor with the key $\overrightarrow{k'}$ such that $\mathcal{L}_p(\overrightarrow{k'} - \overrightarrow{q}) < \mathcal{L}_p(\overrightarrow{k} - \overrightarrow{q})$. Therefore, along the forwarding path of the query, the distance between the key at the current node and the target key $\overrightarrow{q}$ is monotonically decreasing as the query is forwarded. Besides, the probabilistically selected neighbors act as long jumps that ensure exponential decrease of this distance on average. Thus, the average forwarding path length is logarithmic to the size of the network.

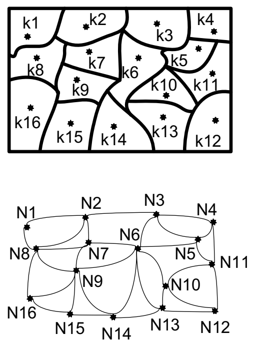
The way we defined the neighborhood relationship between the PDN nodes based on the distance between their keys, together with the clustering property of the resulting small-world topology allow for the effective execution of other types of similarity queries as well. On one hand, we defined the neighborhood relationship such that neighbors of a node have keys closely similar to the key of the node, and consequently, similar to each other. On the other hand, due to the clustering property of the generated small-world graph, neighbors of a node are closely connected in terms of the hop-count in the network (i.e., number of the edges on the path between each pair of nodes). Therefore, a *locality* of tightly connected nodes with closely similar keys is created at the neighborhood of each node in the network. With a topology constructed out of such localities, range and $\mathbf{k}$NN queries can be executed efficiently in two phases, first, by an exact-match query to locate the locality of the query key $\overrightarrow{q}$, and second, by flooding the query throughout the locality of $\overrightarrow{q}$. With a localized topology, flooding at the locality of the query key is efficient. We can locate all the keys relevant to the range and $\mathbf{k}$NN queries in a limited number of hops h away from $\overrightarrow{q}$, where h is independent of the size of the network $|N|$. With our simple PDN example, for range and $\mathbf{k}$NN queries all the relevant keys (and almost only relevant keys) are visited within $h = O(r)$ and $h = O(\lceil \mathbf{k}^{\frac{1}{d}} \rceil)$



a. Recursive partitioning



b. Recursive partitioning example: GNA



c. Flat partitioning

Figure 3: Partitioning of the key space

hops from $\overrightarrow{q}$, respectively. Therefore, for both types of queries, $\mathbf{T}$ is $O(\log|N|+h)$, $\mathbf{C}_1$ is $O(\log|N|+h^d)$, and $\mathbf{C}_2$ is $O(d\log|N|+h^d)$.

With an inclusive key space $V \subset \mathbb{Z}^d$, the simple PDN example considered here is only of illustrative significance. We, however, use the same properties to develop SWAM that applies to more general PDN models.

4.2 SWAM Family

Almost all the traditional access methods for database systems are based on one core idea to reduce the search space for efficient access (see the unified model by Chavez et al. (2001)). They recursively partition the key space into a set of disjoint *similarity* classes[3]. An index is then constructed as a hierarchy of the class representatives at successive levels (see Figure 3-a). The hierarchical index allows filtering out (i.e., to dismiss without inspection) the irrelevant/dissimilar classes while query is directed from the root of the hierarchy toward the similarity class of the query key. The average query time is logarithmic to the size of the database.

By mapping each node of the hierarchy to a PDN node, the same idea can be directly applied to index PDNs, although as we show later the resulting distributed hierarchical index structure is not appropriate for PDNs. Consider K as the set of keys available in a PDN. Any similarity-based relation can be used to partition the key space. For example, in Figure 3-b, V is recursively partitioned based on the GNA approach (Brin, 1995). Starting from V as the global similarity class, at each level the parent similarity class c with the class representative $\overrightarrow{k} \in K$ is partitioned into a set of h disjoint subclasses c_i with representative keys $\overrightarrow{k_i} \in K$ ($i \in I_h = [1..h]$) such that $c_i = \{\overrightarrow{k'} \in V | \mathcal{L}_p(\overrightarrow{k'} - \overrightarrow{k_i}) < \mathcal{L}_p(\overrightarrow{k'} - \overrightarrow{k_j}), \forall j \neq i\}$. Considering that in a PDN each key $\overrightarrow{k}$ resides at a PDN node n_k, the GNA-tree corresponding to such a space partitioning is a *distributed* GNA-tree in which $\mathcal{A}(n_k) = \{n \in N | n = n_{k_i}, i \in I_h\}$. Query processing with such a distributed index tree is similar to that of its corresponding centralized counterpart, with query actually traversing a physically constructed tree rather than a tree structure in memory. Although this indexing approach may seem appealing, due to the lack of a balance load among its nodes, is inappropriate for PDNs. The unbalance load is evident by observing that nodes which represent larger similarity classes (i.e., nodes at the higher levels of the hierarchy) receive more queries to process. In the extreme case, the root of the hierarchy processes all queries. Besides, hierarchical structures are loop-free and intolerant to failures and/or autonomous behaviors of the PDN nodes.

SWAM also employs the space partitioning idea; however, to avoid the problems with hierarchies, instead of recursive partitioning assumes a flat partitioning (see Figure 3-c). Each key $\overrightarrow{k} \in K$ (or $n_k \in N$) represents its own similarity class $c_k \subseteq V$ and the set of $|K|$ similarity classes are collectively exhaustive $V = \bigcup_{k \in K} c_k$ and mutually exclusive $c_k \cap c_{k'} = \varnothing$, $\overrightarrow{k} \neq \overrightarrow{k'}$. An uncharacteristic case is where two or more nodes store replicas of the same key $\overrightarrow{k}$. We assume all such nodes represent the same class c_k redundantly. Such a partitioning scheme can potentially balance the query processing load among PDN nodes. With hierarchies, neighborhood relationship between a pair of nodes is directly derived from parent-child relationship between their corresponding similarity classes to reflect the similarity between their classes. Similarly, with flat partitioning we define the neighborhood relationship based on

[3]The generic mathematical term for *similarity* class is *equivalence* class. Here, the equivalence relations that partition the space are based on the distance (or *similarity*) between the keys.

the adjacency relationship between the similarity classes $\mathcal{A}(n_k) = \{n_{k'} \in N | c_k \text{ and } c_{k'} \text{ are adjacent}, k' \in K\}$. The resulting index structure is a graph instead of a loop-free tree. Besides, processing of the query can start from any node (e.g., the actual query originator) rather than exclusively from a unique node, the root.

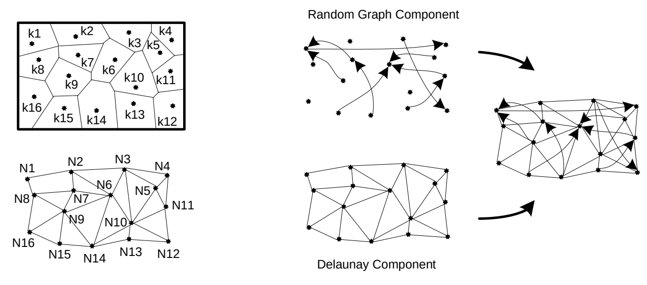
The challenge is to define the similarity-based partitioning relation such that the resulting graph-based index structure bears indexing characteristics similar to those of the hierarchical index structures. Particularly, it should allow filtering of (i.e., avoid visiting) the irrelevant classes effectively as query is directed from a query originator toward the similarity class of the query key. Moreover, to support range and **k**NN similarity queries effectively, alike hierarchical index structures similar classes should be in proximity of each other in terms of the hop-count in the index topology. Finally, the $O(\log N)$ expected query time achieved by the hierarchies is also desirable with the graph-based index structure. As outlined in Section 4.1, these requirements are addressed by the properties of a basic small-world graph. A SWAM index structure is a general graph-based index structure that satisfies a generalization of the same properties as follows:

Property 1 : Monotonic approach toward query key When a node with key $\overrightarrow{k}$ receives a query $\overrightarrow{q}$, always either $\overrightarrow{q} \in c_k$, or the node has at least one neighbor with a key $\overrightarrow{k'}$ such that $\mathcal{L}_p(\overrightarrow{k'} - \overrightarrow{q}) < \mathcal{L}_p(\overrightarrow{k} - \overrightarrow{q})$. Consequently, if the node n_k receives the query $\overrightarrow{q}$, it is guaranteed that for all $\overrightarrow{k''} \in \{\overrightarrow{j} \in K | \mathcal{L}_p(\overrightarrow{j} - \overrightarrow{q}) \geq \mathcal{L}_p(\overrightarrow{k} - \overrightarrow{q})\}$ the node $n_{k''}$ will never be visited in future during the greedy forwarding, and the similarity class $c_{k''}$ is filtered.

Property 2 : Localized index topology With a localized index, for each node n_k the set of nodes at its neighborhood $\mathcal{A}(n_k)$ are tightly connected and store keys closely similar to $\overrightarrow{k}$. We measure these two characteristics with the two metrics Clustering Coefficient (CC) and Neighbor Distance Distribution (NDD), respectively. For a node n, CC$_n$= $C(n)$ is defined by Equation 1. For a graph G, CC$_G$= $\frac{1}{|N|} \sum_{\forall n \in N}$ CC$_n$. Also, NDD is the probability distribution function of the random variable $\overline{X} = \mathcal{L}_p(\overrightarrow{k'} - \overrightarrow{k}), \forall n_k \in N \, \forall n_{k'} \in \mathcal{A}(n_k)$. As we discussed in Section 4.1, a localized topology allows efficient processing of the range and **k**NN similarity queries.

Property 3 : Logarithmic forwarding-path length For an exact-match query (processed by greedy forwarding), on average $\mathbf{T} = O(\log N)$.

Any graph-based index structure that maintains these SWAM properties is a member of the SWAM family. In Section 4.3, we introduce an example SWAM index structure.



a. Voronoi diagram and dual Delaunay graph

b. SWAM-V topology

Figure 4: SWAM-V index structure

4.3 SWAM-V: A Voronoi-based SWAM

SWAM-V partitions the key space V to a Voronoi diagram (Okabe et al., 2000) (see Figure 4-a). For each key $\overrightarrow{k_i} \in K$ ($i \in I_{|K|}$), $n_{k_i} \in N$ represents the similarity class $c_{k_i} = \{\overrightarrow{k} \in V | \mathcal{L}_p(\overrightarrow{k} - \overrightarrow{k_i}) < \mathcal{L}_p(\overrightarrow{k} - \overrightarrow{k_j}), \forall j \neq i\}$, which is the *Voronoi cell* of n_{k_i}. Accordingly, the neighborhood of the node n_{k_i} is defined as $\mathcal{A}(n_{k_i}) = \{n_{k_j} \in N | c_{k_i} \text{ and } c_{k_j} \text{ are adjacent}, \forall j \in I_{|K|}\}$. Nodes that store replicas of the same key share the same neighborhood; i.e., if $\overrightarrow{k_i} = \overrightarrow{k_j}$, $\mathcal{A}(n_{k_i}) = \mathcal{A}(n_{k_j})$. The resulting graph is the dual Delaunay graph of the Voronoi diagram and is unique for each diagram (see Figure 4-a). Since the neighborhood relationship is symmetric, the Delaunary graph is depicted as an undirected graph. The SWAM-V topology consists of a random graph component (identical to that of the small-world graph) that is superimposed over the Delaunay graph (see Figure 4-b).

Theorem 1. *The SWAM-V index structure satisfies the SWAM Properties 1, 2, and 3.*

More details about SWAM-V, including the SWAM-V join primitive (for index construction) and forward primitives (for query processing) is presented in Banaei-Kashani and Shahabi (2004).

5 CONCLUSIONS

In this paper, for the first time we introduced and applied the complex system theory as a modelling framework for PDNs. We demonstrated the usefulness of this modelling framework by proposing an efficient search mechanisms for indexable PDNs inspired by the small-world model originally introduced to explain efficient communication in social networks. Specifically, we first defined a formal framework to study the problem of similarity-search in PDNs. Subsequently, we proposed a set of properties to generate efficient index structures (i.e., PDN topologies) for processing similarity queries in PDNs. These properties are realized by a family of access methods, the *SWAM* family. We introduced *SWAM-V*, a member of the SWAM family, which supports exact-match, range, and

kNN queries for PDNs with multi-attribute objects. Leveraging on the SWAM properties, SWAM-V achieves query time, communication cost, and computation cost logarithmic to the size of the network. Moreover, since unlike DHTs, SWAM-V does not enforce the placement of the objects within the network, it avoids unnecessary content replacement, supports object replication, and adapts to the object distribution.

In the short term, we intend to extend the study of our search mechanisms for indexable PDNs by investigating other members of the SWAM family that as compared to SWAM-V enforce less constraining assumptions to the PDN society. For example, currently we are studying *SWAM-P*, an enhanced version of SWAM-V with probabilistic index topology and flexible neighbor selection policies. SWAM-P not only lets PDN nodes maintain their own data, but also allows them to exercise their autonomy in choosing their neighbors. Our initial results show that as the PDN nodes exercise more autonomy, the efficiency of SWAM-P gracefully degrades from that of SWAM-V to the efficiency of sequential scan (Banaei-Kashani and Shahabi, 2003).

In the long term, we follow our twofold research agenda discussed in Section 1. First, we investigate applicability of other models from the complex system theory (either from social networks or other complex systems) to model and address search and other PDN problems. Particularly, we are interested in modelling PDNs on the capitalistic and socialistic human societies with free-market and planned-market, respectively, and compare the efficiency of data transactions and query processing in capitalistic and socialistic PDNs using the existing mathematical economy models for these forms of government. Second, we believe PDNs provide a unique opportunity for the scientists to test and verify their theories about natural complex systems using the PDN replica of those systems as testbed. We intend to advertise this opportunity by providing example testbeds for study of molecular networks.

ACKNOWLEDGEMENTS

This research has been funded in part by NSF grants EEC-9529152 (IMSC ERC) and IIS-0238560 (PECASE), unrestricted cash gifts from Microsoft, and an on-going collaboration under NASAs GENESIS-II REASON project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

1 Albert, R. and Barabasi, A. (2000) ‘Topology of evolving networks: Local events and universality’, *Physical Review Letters*, Vol. 85, pp. 5234–5237.

2 Banaei-Kashani, F. and Shahabi, C. (2003) ‘Searchable querical data networks’, *Proceedings of the International Workshop on Databases, Information Systems and Peer-to-Peer Computing in conjunction with VLDB'03*, Berlin, Germany, September 9–12.

3 Banaei-Kashani, F., Shahabi, C. (2004) ‘SWAM: A Family of Access Methods for Similarity-Search in Peer-to-Peer Data Networks’, *ACM Thirteenth Conference on Information and Knowledge Management (CIKM'04)*, Washington DC, November 8–13.

4 Bar-Yam, Y. (1997) *Dynamics of Complex Systems*, Westview Press.

5 Brin, S. (1995) ‘Near neighbor search in large metric spaces’, *Proceedings of the 21th International Conferenceon Very Large Data Bases (VLDB'95)*, Zurich, Switzerland, September 11–15.

6 Chavez, E., Navarro, G., Baeza-Yates, R.A. and Marroquin, J.L. (2001) ‘Searching in metric spaces’, *ACM Computing Surveys*, Vol. 33, No. 3, pp. 273–321.

7 Doan, A., Domingos, P. and Halevy, A. (2001) ‘Reconciling schemas of disparate data sources: A machine-learning approach’, *Proceedings of ACM International Conference on Management of Data (SIGMOD'01)*, Santa Barbara, California, May 21–24.

8 Kleinberg, J. (2000) ‘The small-world phenomenon: an algorithmic perspective’, *Proceedings of the 32nd ACM Symposium on Theory of Computing*, Portland, Oregon, May 21–23.

9 Ohira T. and Sawatari, R. (1998) ‘Phase transition in a computer network traffic model’, *Physical Review E*, Vol. 58, pp. 193-195.

10 Okabe, A., Boots, B., Sugihara, K. and Chiu, S. (2000) *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, John Wiley, 2nd edition.

11 Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S. (2001) ‘A scalable content-addressable network’, *Proceedings of ACM SIGCOMM '01*, San Diego, California, August 21–23.

12 Rowstron, A. and Druschel, P. (2001) ‘Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems’, *Proceedings of ACM International Conference on Distributed Systems Platforms (Middleware'01)*, Heidelberg, Germany, November 12–16.

13 Sornette, D. (2000) *Critical phenomena in natural sciences: chaos, fractals, selforganization and disorder*, Springer.

14 Stoica, I., Morris, R., Karger, D., Kaashoek, M. and Balakrishnan, H. (2001) ‘Chord: A scalable peer-to-peer lookup service for internet applications’, *Proceedings of ACM SIGCOMM '01*, San Diego, California, August 21–23.

15 Watts, D. and Strogatz, S. (1998) ‘Collective dynamics of small world networks’, *Nature*, Vol. 393, pp. 440–442.