Applied GIS

ISSN: 1832-5505
Vol-10 Issue-01 Jan 2022

A Reliable method for structured P2P network query routing

K NAOMI JYOSTMA, Assistant Professor, JYOSTNA.KONDA45@GMAIL.COM
TALARI SIVALAKSHMI, Assistant Professor, shivalakshmidinesh@gmail.com
KUMMARA RANGA SWAMY, Assistant Professor, rangaswamy.kumara@gmail.com
Department of CSE, Sri Venkateswara Institute of Technology,
N.H 44, Hampapuram, Rapthadu, Anantapuramu, Andhra Pradesh 515722

ABSTRACT—
It may be rather difficult to locate a document or resource in an unstructured peer-to-peer network. In this paper, we present a query routing method that can handle any kind of overlay topology, with nodes having different processing capacities (which can be a reflection of their altruism level, for example) and different class-based likelihoods of query resolution (which can be a reflection of query loads and the distribution of files and resources on the network, among other things). A grade of service constraint, which ensures that the routes of inquiries meet pre-specified class-based limitations on their associated a priori query resolution probability, is used to demonstrate that the approach stabilises query demand. We clearly characterise and statistically compare the capacity region of such systems to that of random walk-based searches. Additionally, the performance advantages in terms of mean delay for the recommended technique are shown by findings from the computational model. We also look at other ways to simplify things, determine parameters, and adjust to different class-based query resolution probabilities and traffic volumes. This index includes concepts such as peer-to-peer, search, stability, backpressure, and random walk.

I. INTRODUCTION

File sharing, video streaming, expert/advice sharing, sensor networks, databases, and other services are just a few examples of the many growing and varied applications of peer-to-peer (P2P) systems. Resolving queries or finding files/resources effectively is one of the fundamental functionalities of such systems. In this work, the issue is discussed. For example, see [1]–[10] for a selection of publications delving into the topic of effective search/routing mechanism design in both organised and unstructured P2P networks. Peers, data, and resources in structured networks are arranged in overlays with specified structures and characteristics. For example, you may get excellent forwarding-delay qualities by designing search methods that resolve names using distributed hash table (DHT) coordinate systems (see [2]). The key assignment process in such systems could affect the query throughput. Keys to peers and data/service objects must be assigned proactively or reactively for load balancing to operate, as shown in [11], and network hierarchies may need to be exploited [10]. To ensure fast query resolution, especially in dynamic situations with peer/content turnover or when reactive load balancing is needed, the fundamental challenge in such networks is not search/discovery, but rather preserving the structural invariants. In contrast, unstructured networks are less complicated to set up and keep running, but efficient search results are harder to get due to the prevalence of ad hoc overlay topologies in these networks. Each node in an entirely decentralised P2P network is only aware of its immediate overlay neighbours. Unstructured network search

ISSN: 1832-5505
Vol-10 Issue-01 Jan 2022

methods have relied on limited-scope floods, simulated random walks, and variations thereof due to the scarcity of available data [3]-[5]. A lot of studies in this field have looked at how various search methods perform in terms of contact time, or the amount of hops needed to locate the target.

refer to, for instance, [4]-[6], which deal with the spectral theory of Markov chains on (random) graphs. Sadly, these search approaches don't hold up well under heavy query loads in heterogeneous environments where peers' service capacities or resolution likelihoods differ. The shortcomings of completely unstructured networks may be somewhat remedied by hybrid P2P systems, such as Gnutella and FastTrack. In these types of networks, there is a straightforward two-tiered hierarchy in which some peers act as "super-peers." In a hub-and-spoke configuration, these high-level nodes are well-connected to both other super-peers and a group of lower-level nodes [12]. The suggested search strategies continue to rely on variations of floods and random walks, even if these systems provide scalability benefits. Using information from reverse-path forwarding, the authors of [7] suggest a system in which peers store the results of previous requests in a cache. The goal is to intelligently "bias" their forwarding choices by connecting query classes with neighbours who can best resolve them based on prior experience, thereby learning the optimum method to send certain classes of requests. Unfortunately, there are a lot of moving parts with this method, and it isn't load sensitive or performance guaranteed just yet.

.

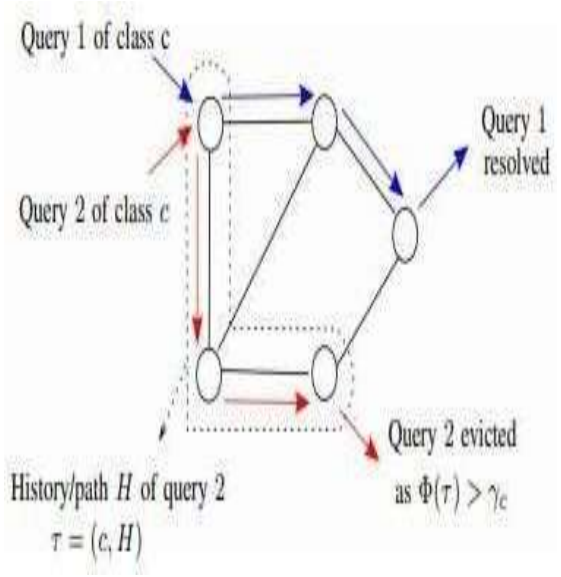


Fig. 1. A network of super-peers G = (N , L). Queries of a given class traverse potentially different routes. A query eithergets resolved or gets evicted from the network upon receiving a

ISSN: 1832-5505
Vol-10 Issue-01 Jan 2022

grade of service.

1. It dynamically accounts for heterogeneity in super-peer's 'service rate,' reflecting their altruism, and query loads across the network. To the best of our knowledge, this is the first work to rigorously account for such heterogeneity in devising a search mechanism for P2P networks.

2. It is based on classifying queries into classes. This classification serves as a type of name aggregation, which enables nodesto infer the likelihoods of resolving classqueries, which, in turn, are used in learning how to forward queries.

3. Our approach is fully distributed in that it involves information sharing only amongst neighbors, and achieves stability subject to a Grade of Service (GoS) constraint on query resolution. The GoS constraint corresponds to guaranteeing that each query class followsa route over which it has a reasonable 'chance' of being resolved.

4. We provide and evaluate several interesting variations on our stable mechanism that help significantly improve the delay performance, and further reducethe complexity making it amenable to implementation. Specifically, we formally show that backpressure with aggregated queues, where aggregation is based on queries' histories, is stable for fullyconnected super-peer networks. This provides a basis for substantially reducing complexity by approximations, e.g., in the case where content is randomly placed. Organization. In Section II, we set up our basic system model. We characterize the stability region of the network and provide the stable protocol and several modificationsin Section III. We provide some numerical results in Section IV. We discuss estimation of query resolution probability and ways to reduce implementation complexity inSection V.

II. SYSTEM MODEL

It is assumed that the overlay linkages, denoted as L ∪ N →N, are symmetrical, meaning that if (i, j) 2 L, then (j, i) 2 L. The directed graph G = (N,L) represents the overlay network, with N nodes representing the super-peers.

Our collection of neighbours of super-peer i is denoted as N(i). Peers lower on the hybrid network's hierarchy are not shown directly but are instead linked to the super-peer they are a part of. The service rate μi, which is related with the positive integer number of queries that each super-peer i is willing to settle or transmit in each slot, is based on the assumption that time is slotted. Our premise is that superior peers maintain a log of the resources and files accessible to inferior peers. When a subordinate peer becomes a super peer, this knowledge is relayed to their superior peers. While subordinate peers may approach a super peer with a query, they are not allowed to take part in the forwarding or

ISSN: 1832-5505
Vol-10 Issue-01 Jan 2022

querying processes. final decision Let C represent a collection of preset resource classes and R represent the set of all files and resources that could be searched over the network. The files or resources that correspond to class c are denoted as Rc ⇢ R for every c 2 C. For every c in C and every i in N, let Rc i denote the collection of class c files and resources accessible at super-peer i or its subordinate peers. The number of inquiries that reach super-peer i or its subordinates at time t is represented by the random variable Ai(t), and the chance that a query is for

r 2 R is the file/resourcename. When a query's target resource is in Rc, we call it a class c query. The number of classes c is represented by Ac i (t). questions that come to super-peer i or its subordinates at time t. We have clearly defined arrival rates, (c i : 8i 2 N, c 2 C), where c i is the mean arrival rate of class c queries at node i, since we assume these random variables are rate ergodic, independent across slots, and have limited second moments. It is possible for node i to transmit a class c query to a neighbouring node if it cannot be resolved locally. Each node's class and history—the collection of nodes it has visited before—contribute to the probability that it can answer such a query. The history is presented in an unorganised manner. Assume, for the sake of argument, that three nodes in a network divide up the class c-related files and resources (Rc). The third node will definitely resolve a class c query if the first two nodes tried and failed. In different settings, a decreased conditional chance of resolution at the next node may be seen if a search for a certain media file fails at many nodes; this is because the file is probably uncommon.

III. STABLE QUERY FORWARDING POLICY

In this section, we will propose a query scheduling and forwarding policy that ensures the GoS for each class, is distributed, easy to implement, and is stable.We begin by defining the stability for such networks and the associated capacity region.

A. Stability & Capacity Region We shall use the definition of network stability given in [14], which is general in that it includes non-ergodic policies. However for ergodic policies it is equivalent to standard ofnotions of stability given in [13], [22]. For a given queue process Q☒ i (t), let g☒ i (↵) denote its ‘overflow’ function

Definition 1. A queue Q☒ i (t) is stable if g☒ i (↵) ! 0 almost surely as ↵ ! 1. The network is stable if each queue is stable. Next we define the ‘capacity region’ for query loads on our network. Definition 2. The capacity region *T* is set of query arrival rates , such that there is a feasible solution to the following linear constraints on f , f ☒ ij :(i, j) 2 L, ☒ 2 T : Capacity constraints: for all i 2 N

We refer to f as flow variables, where (4) ensures that the incoming flow to a node is less than its service rate and (5) ensures that the total flow of types ☒ 0 2 E1 i (☒) reaching i which is not resolved at i (left hand side) equals the flow of type ☒ leavingnode i. These are different than the standard

ISSN: 1832-5505
Vol-10 Issue-01 Jan 2022

multicommodity flow conservation laws in the sense that our conservation equations are designed to capture the following aspects arising in P2P search systems: (a) history dependent probability of query resolution at each node, (b) updates in 'types' of queries as they get forwarded to different nodes, (c) computing the quality of service received by query via its history and designing an appropriate exit strategy upon receiving enough service. Recall, $T0$ denote the interior of T. The following theorem proved in Appendix A makes the link between the capacity region and stabilizability of the network.

Note that this result is general in that even full knowledge of future events does not expand the region of stabilizable rates. Also, while our focus, for now, is on policies where p corresponds to the conditional probabilities of query class resolutions, subject to the GoS modification, other modifications could be made. The only restrictions on p for above result is that each query should eventually leave the network, and revisits to nodes (while allowed) have a zero probability of resolving the query

B. Stable policies In principle, given 2 $T0$, a feasible set of network flows can be found and, as shown in the proof of Theorem 1.b, this can be used to devise a fixed randomized policy which stabilizes the network. However, such a centralized policy

may not be practically feasible, moreover arrival rates may not be known a priori. Further, designing a stable search algorithm is now a challenge since, while the routing decisions are to be based on instantaneous queue loads at the neighbors, the decisions themselves affect the type/queue to which a query belongs. Below we develop a distributed dynamic algorithm where each node i makes decisions based on its queue states and that of its neighbors and only needs to know (or estimate) p☒ i , ☒ 2 T , i.e., local information.

The proof of the above theorem is provided in Appendix B. The proof handles the evolution of query types and the randomness in resolution of queries by incorporating expected queue backlogs into Lyapunav drifts. The basic backpressure algorithm, though stable, is highly wasteful. In a slot, each node i serves only the queue with highest relative backlog. In case that particular queue has less than µi queries waiting in it, the spare services are provided to blank queries, even if the other queues are non-empty. We now devise a more efficient protocol that serves blank queries only when all the queues are non-empty and is thus work-conserving; and is stable as well. As

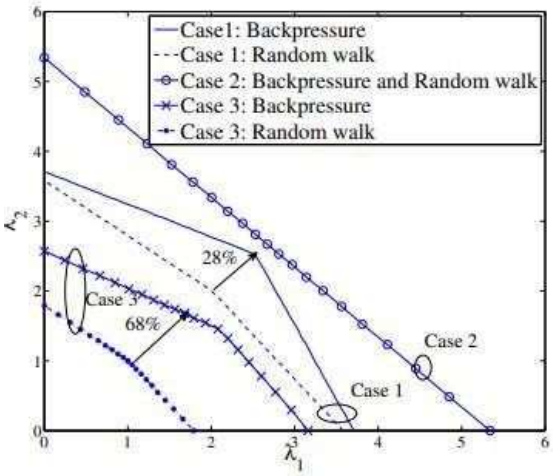
ISSN: 1832-5505
Vol-10 Issue-01 Jan 2022

we shall see, this provides large delay benefits over the above basic backpressurealgorithm. The idea is, if the number of queries in the queue with highest relativebacklog is less than total service rate, the work conserving policy serves the queries insecond highest backlogged queue, and so on,until either total of μi queries are served or all the queues are empty. We formally define the algorithm as follows.

Fig. 2. Boundaries of capacity regions forthe stable backpressure algorithm and random walk policy for the 3 cases.

IV. NUMERICAL RESULTS ANDSIMULATIONS

In this section, we numerically evaluate the gains in the capacity region achievable by our stable backpressure algorithms versus that a baseline random walk policy. We consider a fully connected network with 6 nodes. Let N = {1, 2,..., 6}. Since a super-peer network is designed to be highlyconnected in practice, a fully connected network might be a good representative of the practice. We consider two query-classes, c1 and c2. We assume that arrival rates for a given class is same at all the nodes, say 1 forclass c1 and 2 for class c2. This reduces the dimension of the capacity region from 12 to 2, making it easier to study. Further, the parameters for the GoS, viz., c, are set to 0.9for both the classes. In the baseline random walk policy, upon service, each node forwards an unresolved query to one of the neighbors chosen uniformly at random. Since, in a fully connected network, allowing queries to revisit nodes provides no advantages, queries are forwarded to onlythose nodes which are not previously visited. As with backpressure, whose achievable capacity



region is given by Definition 2, we can characterize the achievable capacity region for the random walk policy. It is the set of arrival rates that satisfy the constraints (4)-(6), along with additional constraints that ensure that the outgoing flows of each type at each node areuniformly divided among unvisited nodes. Formally, these are given by,

ISSN: 1832-5505
Vol-10 Issue-01 Jan 2022

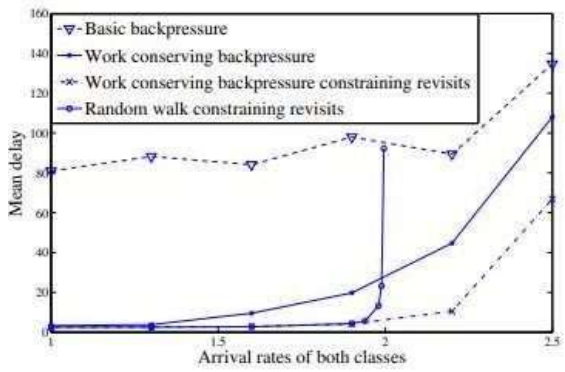


Fig. 3. Delay performance of the backpressure algorithms and random walk for Case 1.

We now compare the delay performance of the backpressure algorithms and random walk under Case 1. Fig. 3 exhibits the mean delay as a function of the arrival rates for both the classes, keeping the arrival rates equal. It confirms our observation that the basic backpressure algorithm is stable, but wasteful as it is not work conserving. The work conserving algorithm significantly improves performance. Performance is further improved by constraining queries from revisiting nodes. With this modification, the backpressure algorithm has excellent delay performance as compared to the random walk policy with the same revisit constraints and same GoS, especially at higher loads

V. IMPLEMENTATION ANDCOMPLEXITY

A. Estimating query resolution

probabilities So far we have assumed that resolution probabilities for queries of different types are known. In practice they can be easily estimated. In order to ensure unbiased estimates can be obtained at each node, suppose a small fraction ✏ of all queries is marked 'RW', forwarded via the random walk policy with a large TTL, and given scheduling priority over other queries. With a sufficiently large TTL this ensures that each node will see a random sample of all query and types it could see and thusallow for unbiased estimates. All queries which are not marked 'RW' are treated according to our backpressure policy based on the estimated query resolution probabilities. A node i receives O(t✏) 'RW' marked samples in time t. Thus, standard deviation in the estimation error is O(q 1

✏t). Thus the error is small for large enough t. If the contents are static, one may discontinue the estimation process afterlarge enough time t, in which case the time-averaged performance of the policy remains unchanged. Alternatively, to allow persistent tracking of changes in resolution probabilities, we may estimate the queryresolution probabilities via samples providedfrom a control algorithm, without using aseparate unbiased random walk. The convergence of estimation and stability of the system can be jointly obtained via

ISSN: 1832-5505
Vol-10 Issue-01 Jan 2022

stochastic approximation framework [23] under time scale separation between content dynamics and search dynamics.

B. Reducing complexity Not unlike standard backpressure-based routing our policies suffer from a major drawback: each node needs to share the state of its potentially large number of non-empty queues with its neighbors. For backpressure-based routing the number of queues per node corresponds to the number of flows (commodities) in the network. In our context, the number of queues per node corresponds to number of query types it could see, i.e., worst case →ı(|C|2|N |). In this section we propose simple modification and approximations that considerably reduce the overheads, albeit with some penalty in the performance. The key idea is to define equivalence classes of query types that share a 'similar' history, in the sense that theyhave similar conditional probabilities of resolution, and have them share a queue. For example, all query types of class c which have visited the same number of nodes k might be grouped together, reducing the queues to →ı(|C||N |) or better.Alternatively we will show one can further reduce overheads by approximately grouping similar query types based on their classes c and the cumulative number of classc files/resources they have seen in nodes in H(☒), reducing the number of queues to →ı(|C|L) where L is a set of quantization levels. Intuitively such queries have seen similar opportunities if files/resources are randomely made available in the network.

VI. CONCLUSION

Our unique, decentralised, and trustworthy search strategy for super-peer-enabled unstructured P2P networks was the main point of our presentation. Capacity improvements of up to 68% compared to conventional random walk methods are possible with our backpressure-based approach. To make the algorithm more implementable, we also made certain adjustments.

VII. REFERENCES

"Peer-to-peer Wikipedia, free encyclopaedia."
[1] Wikipedia. 2011 (Peer-to-peer), according to Wikipedia.
In the 2003 issue of the IEEE/ACM Transactions on Networking, the authors Stoica et al. present "Chord: a scalable peer-to-peer lookup protocol for internet applications" (vol. 11, no. 1, pp. 17–32).
[3] "Searching techniques in peer-to-peer networks" by X. Li and J. Wu was published in 2004 by CRC Press in the Handbook of theoretical and algorithmic aspects of ad hoc, sensor, and peer-to-peer networks. The book was edited by Wu and published in Boca Raton, USA.
"Random walks in peer-to-peer networks," in 2004's Proc. IEEE INFOCOM, was written by **C.** Gkantsidis, M. Mihail, and A. Saberi.
In the 2005 IEEE INFOCOM conference, C. Gkantsidis, M. Mihail, and A. Saberi presented a paper titled "Hybrid search schemes for unstructured peer to peer networks."