Applied GIS

ISSN: 1832-5505
Vol-9 Issue-03 July 2021

A robust approach to routing queries in structured P2P networks

VIJAYA BHASKAR MADGULA, ASSISTANT PROFESSOR, vijaya.bhaskar2010@gmail.com
DIGALA RAGHAVARAJU, ASSISTANT PROFESSOR
raghava.digala@gmail.com
BALAJI SUNIL CHANDRA, ASSISTANT PROFESSOR
hod.cse@svitatp.ac.in
Dept of CSE. Sri Venkateswara Institute of Technology,
N.H 44, Hampapuram, Rapthadu, Anantapuramu, Andhra Pradesh 515722

ABSTRACT—

Documents and resources may be difficult to locate in an unstructured P2P network. This paper presents a query routing method that takes into consideration various factors. These include nodes with varying degrees of altruism, different processing capacities, and different class-based likelihoods of query resolution at nodes. These factors can be influenced by query loads and the distribution of files and resources across the network. We show that this method stabilises query load under a grade of service constraint, which is the assurance that the paths taken by queries meet certain class-based restrictions on the a priori likelihood of their resolution. 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. These words are all indexes: peer-to-peer, search, stability, backpressure, irregular walk.

INTRODUCTION

When it comes to delivering services like file sharing, video streaming, expert/advice sharing, sensor networks, databases, etc., P2P systems are becoming more and more popular due to their distributed nature, scalability, and robustness. Resolving queries or finding files/resources effectively is one of the fundamental functionalities of such systems. This paper deals with this issue. 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

ISSN: 1832-5505
Vol-9 Issue-03 July 2021

surfaces and attributes. To attain excellent forwarding-delay qualities, one might create search methods that conduct name resolution using coordinate systems of distributed hash tables (DHTs), as shown, for example, in [2]. How keys are allocated in such systems might affect the query flow. Consequently, key assignments to peers and data/service objects must be proactive or reactive in order to provide load balancing, as shown in [11], and potentially taking use of network hierarchies [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. However, effective searches are difficult to implement in unstructured networks because to their mostly ad hoc overlay topologies, despite the fact that these networks are easy to set up and maintain. Nodes in completely decentralised P2P networks are only aware of their overlay neighbours. Due to the lack of data, most unstructured network search methods have relied on limited-scope floods, simulated random walks, or variations thereof [3]-[5]. Contact time, or the number of hops needed to locate the target, has been the primary metric for assessing different search algorithms in this field of study. for example, go to [4]-[6] for work on (random) graphs and the spectral theory of Markov chains. Sadly, these search approaches don't hold up well under heavy query loads in heterogeneous environments where peers' service capacities or resolution likelihoods differ. Systems that combine organised and unstructured elements, such as FastTrack and Gnutella2, help to alleviate some of the drawbacks of completely unstructured networks. Some peers act as "super-peers" in these systems, which employ a straightforward two-tiered hierarchy. 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]. Even though these systems are more scalable, the search methods that have been suggested still rely on variations of floods and random walks. According to the research in [7], one way to implement reverse-path forwarding is for peers to 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. There is a lot of extra work involved with this method, and it isn't load sensitive or erformance guaranteed just yet.

Applied GIS

ISSN: 1832-5505
Vol-9 Issue-03 July 2021

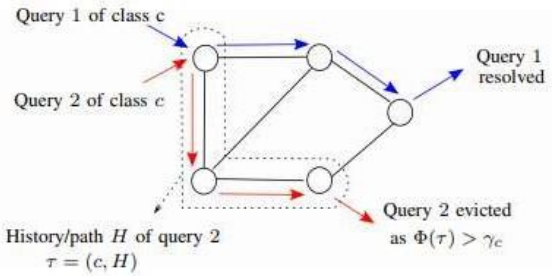


Fig. 1. A network of super-peers G = (N , L). Queries of a given class traverse potentially different routes. A query either gets resolved or gets evicted from the network upon receiving a grade of service.

1. It adapts on the fly to account for variations in the network-wide query loads and the'service rate,' which is a measure of a super-peer's altruism. This is the first study that we are aware of that thoroughly considers such heterogeneity while designing a search method for P2P networks.
2. The foundation is the process of categorising requests. As a kind of name aggregation, this categorization helps nodes learn how to send inquiries by inferring the likelihoods of resolving class queries.
3. Our method is completely decentralised since it achieves stability under a Grade of Service (GoS) limitation on query resolution and only shares information with neighbours. The GoS constraint is equivalent to making sure that every query class adheres to a path that it is more likely to be addressed along.
4. Our stable approach has several intriguing modifications that may be evaluated and used to increase delay performance and make it more implementable by reducing complexity. Our formal proof focuses on fully linked super-peer networks and their stability under backpressure with aggregated queues, where aggregation is based on inquiries' histories. For situations where material is randomly put, for example, this gives a foundation for significantly decreasing complexity via approximations. Settling down. We established our fundamental model of the system in Section II. Section III presents the stable protocol and various variations, while Section IV describes the network stability zone. In Section IV, we provide a few numerical findings. Section V delves into methods for estimating the likelihood of query resolution and strategies for simplifying implementation.

I. SYSTEM MODEL

A directed graph G = (N,L) represents the overlay network, with N nodes representing the super-peers and L ⇓ N →⇥N representing the overlay connections. These links are considered to be symmetric, meaning that if (i, j) 2 L, then (j, i) 2 L as well.

ISSN: 1832-5505
Vol-9 Issue-03 July 2021

This collection of neighbours of super-peer i is denoted as N(i). Keep in mind that in a hybrid network, subordinate peers aren't really represented, but rather linked to the super-peer they're supposed to be working with. Every super-peer i is assigned a service rate μi, which is proportional to the positive integer number of queries it is willing to settle or forward in each slot, assuming that time is allocated in slots. In this case, we'll pretend that super peers log all of the resources and files accessible at each subordinate peer. When a subordinate peer becomes a super peer, this knowledge is relayed to their superior peers. Even though they can't take part in forwarding or querying, subordinate peers may start a query request at a super peer. final decision Let R represent the collection of all network-accessible files and resources and C represent the set of all resource classes that have been declared in advance. Let Rc ⇢ R denote the files/resources associated with class c for every c 2 C. Let Rc i be the collection of class c files and resources accessible at super-peer i or its subordinate peers for every c in C and i in N. For every given time t, let Ai(t) represent the number of inquiries that reached super-peer i or its subordinates, and make a query's likelihood of being resource/file r 2 R. Any query that looks for a resource in Rc is considered 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. Class C queries that fail to be answered at node i may be passed on to a neighbouring node. Along with its class, a node's history—the collection of nodes it has visited in the past—determines the possibility that it can handle such a query. The history is presented in an unorganised manner. Take the hypothetical case of three nodes in a network that share the class c-related files and resources as an example. If a class c query is tried and fails at two of these nodes, it will definitely be resolved at the third node. 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.

STABLE QUERY FORWARDING POLICY

ISSN: 1832-5505
Vol-9 Issue-03 July 2021

Presented here is a distributed, stable, easy-to-implement, and GoS-ensuring query scheduling and forwarding strategy.

We start with a definition of the capacity region and stability for such networks.

Stability & Capacity Region Because it encompasses non-ergodic rules, the definition of network stability provided in [14] will be used. Nevertheless, in the case of ergodic policies, it is the same as the stability standards provided in [13], [22]. The 'overflow' function of a particular queue process, Q☒ i (t), is denoted by g☒ i (↲). Definition 1. A stable queue Q☒ i (t) is one in which the probability that g☒ i (↲)! 0 increases as ↲! 1. If every queue is stable, then the network is stable. The next step is to establish the network's "capacity region" for query loads. In order for the following linear restrictions on f to have a viable solution, the set of query arrival rates that make up the capacity area *T* is defined as follows: The set (i, j) is divided into two parts: L and T. Limited capacity: for every i between 2 and N We call f variables that govern the flow of data the flow of types ☒ 0 2 E1 i (☒) that arrive i and are not resolved at i (left hand side) equals the flow of type ☒ that leaves node i, where (4) guarantees that a node's incoming flow is less than its service rate and (5) states the same thing. Our conservation equations aim to capture the following elements, which differentiate them from the usual multicommodity flow conservation laws: resulting from peer-to-peer search systems: (a) the likelihood of query resolution at each node depending on its history, (b) changes in query "types" as they are passed to different nodes, (c) calculating the quality of service received by query based on its history, and (d) developing an appropriate exit strategy when receiving sufficient service. Remember that *T*0 represents the inside of *T*. Appendix A proves the following theorem, which establishes a connection between the network's stabilzability and its capacity area. Keep in mind that this finding holds true in any scenario where the range of stabilizable rates is not explicitly extended, even when all possible future outcomes are known. Additionally, other adjustments might be done, even though we are now concentrating on policies where p represents the conditional probabilities of query class resolutions, subject to the GoS change. For the aforementioned conclusion, the only constraints on p are that all queries must exit the network at some point and that revisits to nodes, while permitted, have no chance of completing the question.

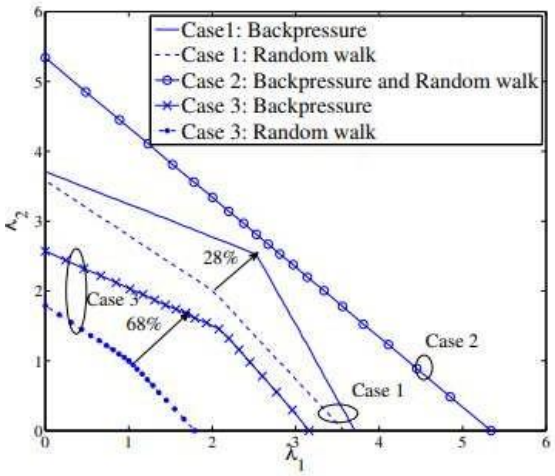
Stable policies Basically, with 2 *T*0, it is possible to identify a workable set of network

ISSN: 1832-5505
Vol-9 Issue-03 July 2021

flows and, as seen in the demonstration of Theorem 1.b, use this information to create a fixed randomised policy that stabilises the network. And yet, such an all-encompassing programme

might not work in practice, and there's no guarantee that arrival rates will be known in advance either. In addition, it is currently difficult to create a reliable search algorithm since routing choices impact the type/queue to which a query belongs, even if they should be based on instantaneous queue loads at the neighbours. In the distributed dynamic algorithm that we have developed below, every node i uses its own and its neighbours' queue statuses to make choices; all it has to know, or estimate, is p☒ i, ☒ 2 T, which is local knowledge.

Appendix B contains the proof of the theorem mentioned before. By including anticipated queue backlogs into Lyapunav drifts, the proof takes care of the development of query types and the unpredictability in query resolution. Although stable, the fundamental backpressure method is quite inefficient. The highest relative backlog queue is the only one that each node in a slot serves. Blank requests are given priority over other non-empty queues if that specific queue has less than half of the queries waiting in it. We have recently developed a more robust and effective technique that utilises work conservation by only serving blank requests when all queues are not empty. Compared to the aforementioned fundamental backpressure, this offers significant delay advantages, as we will show. algorithm. The concept is that the work conserving strategy



will serve the queries in the second most backlogged queue if the total number of queries in that queue is less than the overall service rate. This process continues until either μi queries are serviced or all queues are empty. Here we provide the algorithm's formal definition.

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

II. NUMERICAL RESULTS AND SIMULATIONS

Here, we compare our stable backpressure algorithms to a baseline random walk

ISSN: 1832-5505
Vol-9 Issue-03 July 2021

policy and quantitatively assess the increases in the capacity area that each may achieve. A completely linked network consisting of six nodes is being considered. Set N equal to {1, 2,..., 6}. Due to the fact that a super-peer network is intended to be very

completely linked network may serve as an apt metaphor for the reality of the situation. Two query classes, c1 and c2, are under consideration. First, let's pretend that all the nodes have the same arrival rate for class c1, and second, let's claim that class c2 has a rate of 2. This makes the capacity area simpler to understand by reducing its size from 12 to 2. In addition, for both classes, the GoS parameters (c) are set to 0.9. As soon as a node is serviced, the basic random walk policy has it forward any unanswered queries to a neighbour that it has randomly selected. Queries are directed to nodes that have not been visited before since, in a fully linked network, there is no benefit to letting queries revisit nodes. Similar to how we can characterise the feasible capacity zone for backpressure (defined in Definition 2), we can do the same for the random walk policy. It is the collection of arrival rates that meet the conditions (4)–(6) and other constraints that guarantee that the outgoing flows of each type at each node are evenly distributed among the nodes that have not been visited. These are officially provided by,

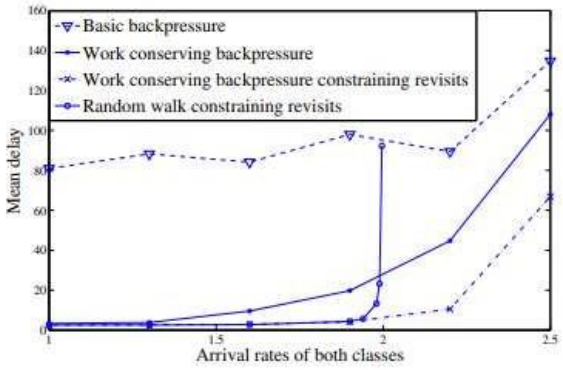


Fig. 3. Delay performance of the backpressure algorithms and random walk for Case 1.

We then evaluate the backpressure methods' and random walk's delay performance in Case 1. Keeping the arrival rates constant, Fig. 3 shows the mean delay for both groups as a function of the rates. It verifies what we suspected: the fundamental backpressure method is stable, but inefficient since it doesn't conserve work. Performance is greatly enhanced by the work saving algorithm. Reducing the number of times queries must visit nodes further boosts performance. This adjustment improves the backpressure algorithm's delay performance relative to the random walk policy under identical revisit

ISSN: 1832-5505
Vol-9 Issue-03 July 2021

limitations and GoS, particularly under heavier loads.

III. IMPLEMENTATION AND COMPLEXITY

Estimating query resolution probabilities So far, we have presupposed that the likelihood of a query's resolution is known across all kinds. They are readily estimable in practice. To make sure that every node may get fair estimates, let's pretend that a tiny percentage of all queries are designated as 'RW,' sent via the random walk policy with a long TTL, and given precedence in scheduling over the rest. By setting the TTL high enough, we guarantee that every node will get an equal number of queries and kinds, allowing us to make fair estimations. When a query is not tagged as "RW," our backpressure strategy is applied depending on the predicted probability of the query's resolution. In time t, a node i gets 'RW' tagged samples at a rate of O(t‒·). This means that the estimating error has a standard deviation of O(q 1 ‒·t). So, for sufficiently big t, the error is negligible. The time-averaged performance of the policy stays the same if the contents are static, allowing one to cease the estimating procedure after a big enough period t. Another option is to estimate the query such that we can persistently monitor changes in resolution probability. resolution probabilities utilising control algorithm samples, without using a separate unbiased random walk separately. Under time scale separation between content dynamics and search dynamics, a stochastic approximation framework [23] may be used to concurrently attain system stability and estimation convergence.

Reducing complexity Just like traditional backpressure-based routing, our rules have one big flaw: every node has to tell its neighbours about the status of its many non-empty queues. There are as many queues per node as there are flows (commodities) in a network that uses backpressure-based routing. If we consider the worst-case scenario, we find that the number of queues per node is equal to the number of query types it is capable of handling, which is equal to half of N. While there will be a performance hit, the overheads will be significantly reduced with the help of our modest modifications and estimates. The main point is that different kinds of queries with "similar" histories (i.e., comparable conditional probability of resolution) should share a queue and be defined as equivalence classes. A such example would be to group together any query types of class c that have visited the same amount of nodes k.

ISSN: 1832-5505
Vol-9 Issue-03 July 2021

This would reduce the as many queues as possible to →(|C||N |) or higher. We will also demonstrate how to further decrease overheads by lowering the number of queues by roughly grouping comparable query types according to their classes c and the total amount of class c files/resources they have viewed in nodes in H(x̄).

→(|C|L) where L is a collection of quantization levels. If files and resources are made accessible in the network at random, such searches have likely encountered comparable chances.

CONCLUSION

Finally, for unstructured P2P networks that include super-peers, we presented a new, distributed, and trustworthy search strategy. When compared to more conventional random walk methods, our backpressure-based strategy may increase capacity by as much as 68%. In addition, we made some adjustments to the method that should make it easier to implement.

IV. REFERENCES

[1] Wikipedia, "Peer-to-peer Wikipedia, free encyclopedia." http://en.wikipedia.org/wiki/Peer-to-peer, 2011.

[2] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," IEEE/ACM Transactions on Networking, vol. 11, no. 1, pp. 17–32, 2003.

[3] X. Li and J. Wu, "Searching techniques in peer-to-peer networks," in Handbook of Theoretical and Algorithmic Aspects of Ad Hoc, Sensor, and Peer-to-Peer Networks, (CRC Press, Boca Raton, USA), 2004.

[4] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," in Proc. IEEE INFOCOM, 2004.

[5] C. Gkantsidis, M. Mihail, and A. Saberi, "Hybrid search schemes for unstructured peer to peer networks," in Proc. IEEE INFOCOM, 2005