Optimization of Assisted Search Over Server-Mediated Peer-to-peer Networks

Zifan He, Leonard Kleinrock
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095, U.S.
zifanhe1202@g.ucla.edu, lk@cs.ucla.edu

***Abstract*—Accompanied by improving accessibility of data and storage and the invention of the blockchain economy, advantages of the peer-to-peer network in privacy and efficiency over server-client systems have become more significant recently. One of the most popular applications of peer-to-peer networks is file transferring, where each peer stores a partial segment of a file and the requester can aggregate each piece into a single file when downloading it. To optimize the performance of file transmission, one goal is to increase the searching speed for the file location over the network. In this paper, we analyze the server-mediated peer-to-peer networks, which is an uncommon architecture mentioned in previous works but can potentially improve the search speed, and minimize the average number of hops during flooding search.**

***Index Terms*—Peer-to-peer network, File transfer, Server-mediated, Configuration model**

I. Introduction

As people's awareness of data security and potential storage failure in both hardware and software rises, backing up personal data from workspaces to disks or the cloud becomes a popular strategy to avoid losing any important file. However, conventional methods of backup have their drawbacks: cloud services such as Google and Dropbox will charge for extra storage and have potential privacy issue, while syncing to a hard disk drive or SSD (RAID stack for NAS) is less flexible and may be broken accidentally. Concerning the scalability and inherited architecture for privacy, a peer-to-peer network provides a favorable internet structure to back up files. People can request drive space from other neighbors when available without being charged. Therefore, the overall utilization of storage hardware would increase. Free software such as *BuddyBackup* [1] has implemented a backup system over an unstructured peer-to-peer network, but it only supports limited operating systems and is not well maintained currently.

The inspiration of this paper is mainly to resolve several theoretical questions regarding the construction of an efficient peer-to-peer (P2P) backup system, including the searching strategy, the file partition, and the optimized protocol for CRUD (create, read, update, delete) operations for future reference when actually implementing one. The remainder of this paper focuses on a specific P2P network architecture, the "server-mediated P2P system", and applies mathematical analysis and simulation to investigate its file searching performance with respect to several hyperparameters and provides a guide to optimize it numerically and analytically. The outcome of the paper can be applied to not only P2P backup system, but also the blockchain systems that are actively investigated recently. Section 3 offers more detail regarding the proposed system design and assumptions about the network, and sections 4 and 5 include mathematical computations of the searching algorithm.

978-1-6654-3540-6/22/$31.00 © 2022 IEEE

II. Background and Related Works

Based on the methodology of organizing neighbors and assisting searching mechanisms, peer-to-peer networks can be categorized into two groups: unstructured and structured. Unstructured peer to peer network protocols, such as Gnutella [4], organizes the network structure by the behavior of each of the peers themselves, where each node maintains a TCP connection with the neighbors of their choice. When searching for a file via flooding, the flooding starts from the requester that crawls through pre-established connections. Structured peer-to-peer networks, such as the IPFS (InterPlanetary File System), may use different kinds of data structures, such as Distributed Hash Tables (DHT) and Merkle DAGs, to identify and locate the peers for better searching performance, while they cannot handle dropping connections as well as an unstructured P2P network [2]. To balance the tradeoffs between structured and unstructured peer-to-peer networks, a common hybridization is to have a server as a mediator to help peers to find each other's files. One of the potential implementations is to reserve several machines as trusted peers and a server will then send the same criteria of the request file to each of them [3]. Since new nodes tend to be clustered around each trusted peer according to the Gnutella protocol, this *assisted search* can potentially increase the searching coverage and improve the searching performance. Although previous research had implemented this architecture using Java Gnutella APIs for experimentations, quantitative analysis for optimizing the model is still missing. There is recent research on blockchain P2P topology with similar network structure, but the main focus is on the network properties and technical implementations, instead of the analysis and optimization [12]. The result of the analysis will be crucial for the actual implementation of an optimized P2P system using

the same structure, such as Spotify [11], to reduce latency and cost. Therefore, this paper takes the server-mediated peer-to-peer system as the basic assumption, with some modifications, and investigates the relationship between file searching speed and several parameters, especially the number of trusted peers and number of replicas of a file in the network.

Due to limited resources of data, a common practice in analyzing an ensemble of peer-to-peer networks is to use a *random graph model* or *configuration model*, so that one can simulate properties of networks to align with reality. Early works on search time over unstructured peer-to-peer networks used the Erdos-Renyi model or the Clustered Erdos-Renyi model, which is a variation of the Erdos-Renyi model by having another probability q for the connections between clusters of the Erdos-Renyi graphs [5]. The results indicate a logarithmic relationship between the number of searching steps and the number of replicas in the network. Uniform probabilistic models are often used to represent the behavior of peer-to-peer networks, numerous communication networks reveal that a better fit is to use power-law degree distribution [6]. One must recognize that switching the architecture and applying it to another model will likely cause the results to change. We choose to use a configuration model with a power-law distributed degree sequence as an interesting and more realistic choice to analyze. For simulation, since we assume the peer-to-peer network as undirected and unweighted, the Barabasi-Albert preferential attachment graph generation will be used [7]. In this model, when a new node is added into the network, the probability that this node will attach to an existing node i is

$$p_i = \frac{k_i}{\sum_j k_j}$$

where k_i and k_j are the degrees of node i and j.

III. System Design

As shown in figure 1, our design is similar to that implemented architecture proposed in the previous work [3], except we add a cache that resides next to the server mediator to store the file identifier and its corresponding replica location in key-value pairs temporarily, with a given timeout, to exploit the temporal locality. The server has the location information of all the trusted peers who themselves send the search requests. Each trusted peer behaves identically as a normal peer, but it is more likely to be attached since they exist at the beginning of the network construction. Trusted peers may open one or more connections with one or more peer clusters. The connections are bidirectional, meaning that both peers on each end of the socket connection can send messages to each other. Trusted peers may also be connected to each other. When a new user executes the network client, it will start an initial search of peers over the network, with a higher preference for the trusted nodes, and will open connections to them.

There are mainly two functions of the trusted nodes. First, since trusted nodes are scattered in multiple peer clusters, having multiple flooding searching starting with these nodes

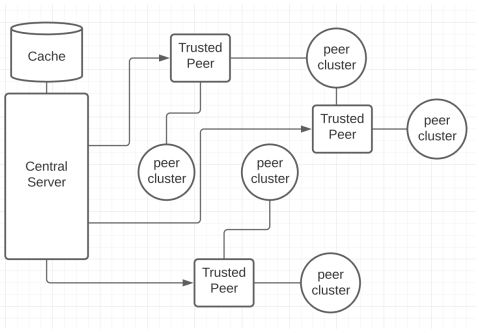

Fig. 1: Architecture of the Network

can increase the possibility of reaching the target file. If the flooding search starts only from the requester, the search will fail if the the requester and the destination are in different connected components in the network. Second, setting up trusted nodes can inherently change the structure of the network. Due to the first mover effect (see Section 13.3 of [7]) applied to trusted nodes, newly joined peers are more likely to connect with them or their neighbors. This infers that even with high dropout rate between peers, there is a high possibility that a connection path between a trusted node and the file destination exists.

When a peer sends a file download request to the server, the server will ask all trusted peers to start a flooding search through all their connections at the same time. The behavior is the same as an unstructured peer-to-peer network. Once the file replica is found by one of the peers, the file will be sent back to the server directly. Finally, the server will open a connection to the requester and send the requested file back to it. In the meanwhile, the file location will be stored into the cache so that if someone requests the same file, it can be retrieved quickly.

For simplicity, during the analysis, the time taken to send a request from the server to trusted peers and the time for the trusted peers to reply to the server will be ignored. These are communications between nodes with known addresses and the time taken has a far smaller scale compared with the time to search the file.

IV. Model Analysis and Approximation

As discussed above, the proposed server-mediated peer-to-peer network will be represented as a configuration model with a power-law degree distribution, meaning that the probability of a node with degree k is [7]

$$p_k = \begin{cases} 0 & k = 0 \\ \frac{k^{-\alpha}}{\zeta(\alpha)} & k \geq 1 \end{cases}$$

where $\zeta(\alpha)$ is the Riemann zeta function and α is a constant between 2 and 3 measured from a real network [7]. In reality, the user would drop out of, or join, the network over time, which may perturb the network structure and thus the degree

TABLE I: Symbols in the Analysis

Label	Meaning
N	The number of nodes in the network
T	Number of trusted peers in the network
c	Average degree of non-trusted nodes
L	Average degree of trusted peers (differ from c since trusted peers are more preferred to be attached)
$\langle k^2 \rangle$	Second moment of degree of nodes
r	Average number of replicas over the network for each file
p_k	Probability that a node has degree k
a_T	number of nodes reachable by all trusted peers
c_n	Number of newly visited nodes covered by the flooding algorithm in the nth iteration.
d_n	Number of nodes left in the network that have not been covered after nth iteration
p	Cost of running a machine as a trusted peer
q	Cost of each hop of searching during flooding over the network
$C(p,q)$	The cost function for searching a file in the peer-to-peer network over time

distribution. For simplicity, we assume that a new node will join the network with the attachment kernel [8]

$$\pi_k = \frac{1}{1-p_0}\frac{\zeta(\alpha)}{\zeta(\alpha-1)}\frac{k^\alpha}{(k+1)^{\alpha-1}}$$

which is the probability that a new node is connected with an existing node in the network. In this case, the overall degree distribution will remain as a power law at any time. Furthermore, since we are flooding through the network and different neighbors may share the same neighbor, the locally tree-like condition will not be assumed in our case, which means we need to compute the overlapping region of nodes' neighbors for each iteration.

For the first iteration of flooding, assume the average degree of all trusted peers is L, which is different c, the average degree of non-trusted nodes over the whole network, since these trusted nodes tend to have higher degrees due to the first mover advantage [7]. Suppose any pair of stubs (the end of an edge that connects to the node) has the same probability to connect and form an edge, then, when we have L nodes attached to the first trusted peer, the fraction of neighbors of the second trusted peer which is not the neighbor of the first trusted peer is $1-L/N$, where N is the total number of nodes. Therefore, the number of neighbors of the first and second trusted peers is

$$a_2 = (1-\frac{L}{N})L + L \tag{1}$$

We can get the number of neighbors of the first n trusted peers as a recursive formula

$$a_n = (1-\frac{a_{n-1}}{N})L + a_{n-1}, \text{ where } a_1 = L \tag{2}$$

To solve the closed form of the recursive relation in (2), we first divide both sides by $(1-L/N)^n$, then let $b_n = a_n/(1-L/N)^n$, and we obtain

$$\sum_{n=1}^{m}(b_n - b_{n-1}) = \sum_{n=1}^{m}\frac{L}{(1-L/N)^n}$$

The right-hand side is a geometric series and the left-hand side will only retain the first and the a_m term. Thus, we can calculate

$$a_n = N(1-(1-L/N)^n) \tag{3}$$

For T trusted peers, the first iteration will cover $a_T = N(1-(1-L/N)^T)$ neighbors.

For the next several iterations, each node will have $D = \frac{\langle k^2 \rangle - c}{c}$ excess degree, the degree of nodes excluding the one from the incoming node, on average, where $\langle k^2 \rangle$ is the second moment of the node degree [7]. Let c_n be the number of nodes covered in the nth iteration and d_n be the number of nodes that have not been visited. At the nth iteration, over D excess degree, Dd_{n-1}/N of them are visited in the previous iteration for large N, and for each of the nodes, some portion of neighbors are visited by the first several nodes. Thus, we have the recursive relationship:

$$c_{n,k} = (1-\frac{c_{n,k-1}}{d_{n-1}})\frac{d_{n-1}}{N}D + c_{n,k-1}$$

where

$$c_{n,0} = \frac{d_{n-1}}{N}D$$

Using a similar method, we can calculate the closed form formula of c_n:

$$c_n = d_{n-1}(1-(1-\frac{D}{N})^{c_{n-1}+1}), \text{ where } c_0 = a_T \tag{4}$$

The number of unvisited nodes is the number of unvisited nodes in the previous iteration less the number of nodes visited in the current iteration, which is

$$d_n = d_{n-1} - c_n, \ d_0 = N - a_T \tag{5}$$

or we can write

$$d_n = d_{n-1}(1-\frac{D}{N})^{c_{n-1}+1} \tag{6}$$

which is a system of recursive relations with two variables. It is hard to find a closed form solution for d_n and c_n. One way is to compute them using a computer program using loops. Another approach is performing an approximation for d_n by inspecting the formula and graph. By numerically computing the first several terms of d_n, we introduce the approximation

$$d_n \approx (N-a_T)(1-\frac{D}{N})^{\sum_i c_i + n} \tag{7}$$

Taking $N = 500$, $D = 3$, $r = 2$, $L = 9$, $T = 3$ as an example, we can numerically calculate $N-d_n$ and d_n and plot them on the graph, shown in figure 2 as the blue and orange curve. Since $c_0 = a_T$, an intuitive guess of approximation of c_n is using a power of n, meaning that

$$d_n \approx (N-a_T)(1-\frac{D}{N})^{a_T n^\alpha + n} \tag{8}$$

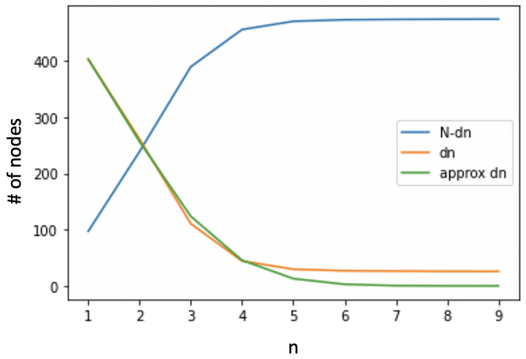

Fig. 2: Plot of $N - d_n$, d_n, and approximated d_n

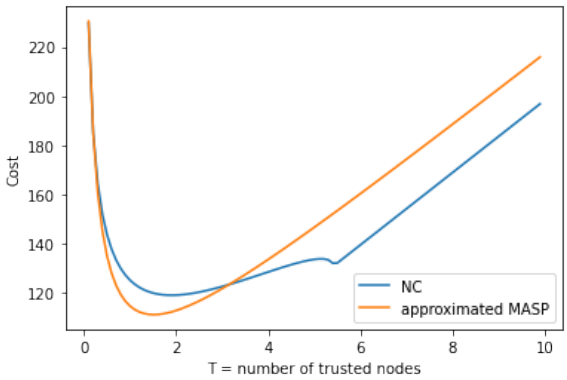

Fig. 3: Plot of cost function for $q/p = 3.49$

for some exponent α. Using numerical fitting by `scipy` on equation (8), the best exponent to fit d_n is $\alpha = 2$, and the fitted line is shown as the green curve.

Let us now assume that the file replicas are distributed uniformly throughout the network. To find the average minimum hops needed to retrieve a file replica, we can apply two types of stopping criterion. The first one checks the fraction of nodes visited in the network, named as node coverage (NC). Since we know the density of file replica, we can know how many nodes we need to visit to find one replica on average. In other words,

$$N - (N - a_T)(1 - \frac{D}{N})^{a_T n^2 + n} \geq \frac{N}{r} \tag{9}$$

where r is the number of replica per file. Taking natural logarithms on both sides and solving the quadratic function of n, we can get the inequality:

$$n \geq \frac{1 - \sqrt{1 + 4a_T \ln \frac{N - N/r}{N - a_T} / \ln(1 - D/N)}}{-2a_T} \tag{10}$$

A second stopping criterion we can use is the minimum average shortest path length (MASP) from all trusted peers. Specifically, we want to find

$$min_{t_1,\ldots,t_T}(E(l_{t_i x}))$$

where t_n is a trusted peer, x is another node in the network, and $l_{t_i}x$ is the shortest path length between t_i and x. If there are r replicas randomly placed in the network for the location assignment, we also only pick the one that is first encountered by the flooding algorithm. Thus, the problem is finding the expectation of the first ordered statistic. The probability density function of $min(x)$ is

$$p(x) = r(1 - F(x))^{r-1} f(x)$$

for a size r random sample. To get the cumulative distribution function and pdf of the original distribution conveniently, we perform a linear approximation on d_n and get $d_n \approx -(N - a_T)((1-(1-\frac{D}{N})^{a_T+1})n - 1)$ if $0 \leq n \leq \frac{1}{1-(1-D/N)^{a_T+1}}$ and $d_n = 0$ otherwise. Subtracting d_n from N we get $N - d_n \approx (N - a_T)(1 - (1 - \frac{D}{N})^{a_T+1})n$ if $0 \leq n \leq \frac{1}{1-(1-D/N)^{a_T+1}}$ and $N - d_n = N$ otherwise. Approximating $c_n = \frac{\partial(N-d_n)}{\partial n}$ and normalizing it, we get the distribution of n on c_n as

$$f_c(x) = \frac{1}{1 - (1 - \frac{D}{N})^{a_T+1}} \tag{11}$$

$$F_c(x) = \frac{1}{1 - (1 - \frac{D}{N})^{a_T+1}} x \tag{12}$$

Therefore, the expected number of hops needed is approximately

$$E[n] \approx r(1 - (1 - \frac{D}{N})^{a_T+1}) \int_0^{1/(1-(1-\frac{D}{N})^{a_T+1})} (1 - (1 - (1 - \frac{D}{N})^{a_T+1})x)^{r-1} x dx \tag{13}$$

where n is the number of hops. Integrating (13) by parts, we find

$$E[n] \approx \frac{1}{(r+1)(1 - (1 - \frac{D}{N})^{a_T+1})} \tag{14}$$

V. Optimization

To optimize the performance, we need to define the target cost function. Define the cost function as

$$C(p, q) = pT + q(W(r, T, N) + 1) \tag{15}$$

where p is the cost in dollar per trusted peer, and q is the cost per second associated with search time of each file, W is the minimum number of hops needed that we obtained in the previous section. Increasing the cost of adding trusted peers provides better performance and whether it is worthwhile depends on developers' value on lower latency. For different cost parameters p and q, the goal is to find the value of T that minimizes the cost $C(p, q)$. To inspect the behavior of the function, two plots are drawn for both approximated MASP and NC with parameters $N = 50$, $L = 6$, $r = 3$, and $D = 3$. As shown in figure 3, the orange curve represents the cost function with the approximated MASP criteria. The blue curve represents the cost function with the node coverage criteria. The y-axis indicates the cost, and the x-axis is the number of trusted peers in the network. The plot is drawn as continuous functions to better observe the trend. In reality,

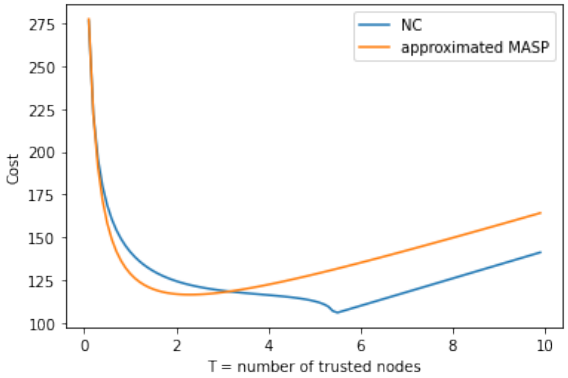

Fig. 4: Plot of cost function for $q/p = 7.75$

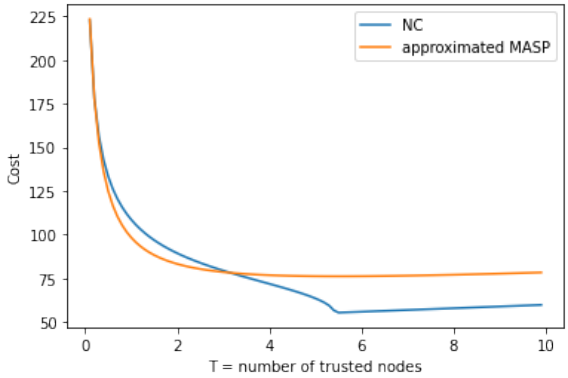

Fig. 5: Plot of cost function for $q/p = 50$

we will only take the integer value of the number of trusted nodes. To some extent, even with the linear approximation, we get similar argmin for both criteria. Due to the complexity of computation, we compute the analytical minimum for the second criteria, which has the cost function

$$C(p,q) \approx pT + q(1 + \frac{1}{(r+1)(1-(1-\frac{D}{N})^{a_T+1})})$$

where $a_T = N(1-(1-L/N)^T)$. Since we are expecting a small T compared with the number of nodes, expanding a_T and $1-(1-\frac{D}{N})^{a_T+1}$ to the first order Taylor expansion is a good approximation. Thus, we get

$$C(p,q) \approx pT + q(1 + \frac{1}{\eta(r,D,N)}) \tag{16}$$

where

$$\eta(r,D,N) = (r+1)(1-(1-\frac{D}{N})(\ln(1-\frac{D}{N})a_T+1)) \tag{17}$$

Taking the derivative with respect to T and solving for the root symbolically using Matlab, we can get that the approximated minimum is at

$$T \approx \frac{\sqrt{\frac{q\ln(\frac{N-D}{D})\ln(\frac{N-L}{N})(N-D)}{p(r+1)}} + \frac{D}{N}}{N\ln(1-\frac{D}{N})\ln(1-\frac{L}{N})(\frac{D}{N}-1)} \tag{18}$$

From figure 3, there is a significant gap between the cost function with and without the linear approximation, since the linear approximation will overestimate the ground truth CDF on the curvy region. This may lead to lower performance of the optimization after approximation for different p and q. With different ratios for $\frac{q}{p}$, the absolute difference between optimized number of trusted nodes (T for which cost is minimized for each curve) for both criteria are different, as shown in figure 4 and 5 (both of which use the same parameter set as for Figure 3).

To investigate the range of the ratio between q and p where both stopping criteria have consensus on the optimal T, let $p = 1$ and q represents the ratio to draw a plot of absolute difference between optimal T for the cost functions of the two criteria with $\frac{q}{p}$ from 1 to 200, shown in figure 6. Since T is an integer in reality, the plot exhibits a staircase behavior. The

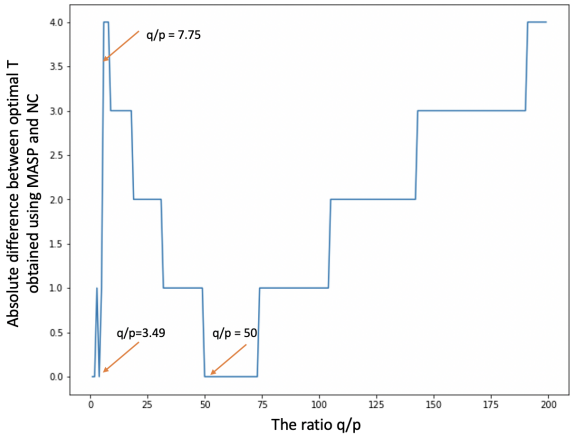

Fig. 6: Absolute difference between optimal T between approximated MASP and NC for $1 \leq \frac{q}{p} \leq 200$, for $N = 50$, $D = 6$, and $r = 3$

cases illustrated on Figure 3 to Figure 5 are critical points of changes for the absolute difference of the optimal solutions between the two stopping criteria.

As the result, if we tolerate the absolute error of optimization on number of trusted nodes within 1, then the average shortest path length stopping criterion will achieve a good performance with either a relatively small q/p ratio (less than 5 in this case) or a medium large q/p ratio ($30 \leq q/p \leq 110$ in this case).

VI. Simulation

The simulation program is based on the Python NetworkX library. To generate a random graph with power law distribution as well as preferential attachment behaviors, we use the function `nx.barabasi_albert_graph()` to create the baseline graph model and remove nodes uniformly at random to simulate the situation when nodes drop off the peer-to-peer network. Furthermore, we also assign the location of file replicas at random, based on the number of replicas we have. Every time the program start a new trial, it will generate a new instance of the random graph with nodes removed, perform a breadth first search, and record the number of iterations needed to find a node with the target file. After finishing all trials, it

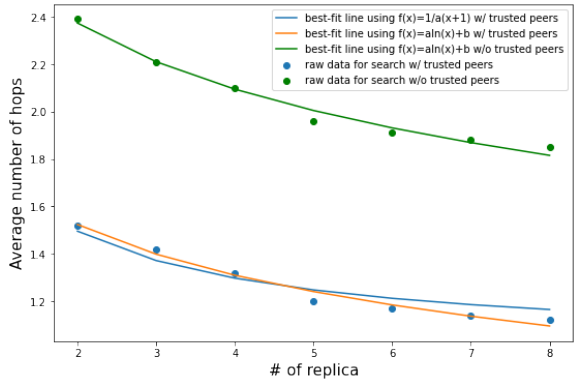

Fig. 7: Average number of hops needed with assisted search

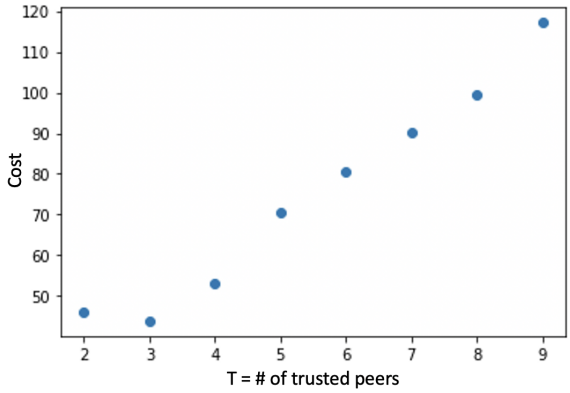

Fig. 8: Cost of finding a file replica against number of trusted peers for $p = 10$ and $q = 10$

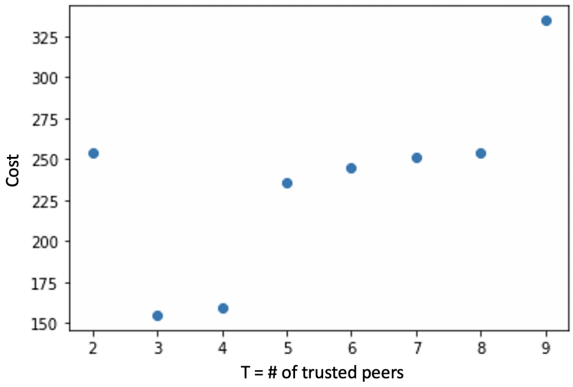

Fig. 9: Cost of finding a file replica against number of trusted peers for $p = 10$ and $q = 90$

calculates the average of the subject of interest and plots the corresponding graph.

For the relationship between number of replica and the number of hops needed to search a file, the simulation initiates an experiment with $N = 500$, $T = 3$, $2 \leq r \leq 8$ for 300 trials. Both cases with or without assisted search by the trusted peers are investigated. If we fit the data points using `scipy.optimize` with the formula of $E[n]$ we found in the model analysis, we get the best fit line as

$$y = \frac{1}{0.665(x+1)} \tag{19}$$

shown as the blue curve in figure 7. This fitted relation is close to our approximation if we plug the average degree of non-trusted nodes $D = 3$ and the total degrees of all trusted nodes $a_T > 100$ in equation (14), indicating that the model analysis in the previous section is reasonable. The orange curve is the best fit line in the logarithm model from previous research [9], with formula $y = -0.315 \ln(x) + 1.75$. Both models have a good prediction for the relationship between the number of replicas and the number of hops for searching a file. The green curve on Figure 7 indicates the same relationship without assisted search (i.e. with no trusted nodes). For every number of replicas, the average number of hops is higher than the case with assisted search, proving an improvement of the performance with trusted peers.

Considering the cost, we ran the experiment for a number of trusted peers. When the cost per machine and cost of searching time has relatively close value, increasing number of trusted nodes will only slightly reduce the cost at the beginning and then escalate linearly, as shown on figure 8. When the cost of running a machine is less than the cost of searching, the total cost will decrease more drastically as the number of trusted nodes increase, as shown on figure 9. These results align with the analytical calculation from the previous sections.

VII. Conclusion

In this paper we derived the relationship between searching time for a file replica, number of trusted peers, and number of replicas over a server-mediated peer-to-peer network with assisted search protocol, and used the cost function to find the optimized setup of number of replicas and number of trusted peers for minimum cost. Unlike previous research which focuses on unstructured peer-to-peer networks with the Erdos-Renyi model, we proposed that the power-law degree distribution is more realistic. Due to difficulty of computing a closed form analytical solution, several approximations are made, while behavior of the cost function still remains as expected: the cost drastically decreases when number of trusted peers increases, but the benefit will diminish, as the number of search hops converges to 1. Once the network builders have the information about the degree of trusted peers, the mean degree, and the total number of peers in the network, an analytical solution can be used for cost reduction, but numerical methods for finding the minimum may provide more accurate results.

Further research will focus on the impact of cache in the cost function, since it will introduce an extra cost on cloud storage correlated with the total size. A higher cache size will increase the hit rate, which reduce the average searching cost. These factors may further change the optimal number of trusted nodes in practice.

Acknowledgment

This research has been supported by Internet Research Initiative (IRI) program direct by Professor Leonard Kleinrock

at University of California, Los Angeles.

REFERENCES

[1] Solutions, NXT Digital. “Buddybackup: Safe, Free and Secure Data Backup.” Buddy Backup, https://www.buddybackup.com/about/how-buddybackup-works.aspx.

[2] Benet, Juan. ”Ipfs-content addressed, versioned, p2p file system.” arXiv preprint arXiv:1407.3561 (2014).

[3] Kwok, Sai Ho, K. Y. Chan, and Yat Ming Cheung. ”A server-mediated peer-to-peer system.” ACM Sigecom exchanges 5.3 (2005): 38-47.

[4] Ripeanu, Matei. ”Peer-to-peer architecture case study: Gnutella network.” Proceedings first international conference on peer-to-peer computing. IEEE, 2001.

[5] Tewari, Saurabh, and Leonard Kleinrock. ”Optimal search performance in unstructured peer-to-peer networks with clustered demands.” IEEE Journal on Selected Areas in Communications 25.1 (2007): 84-95.

[6] Adamic, Lada A., et al. ”Search in power-law networks.” Physical review E 64.4 (2001): 046135.

[7] Newman, M., *Networks*, 2nd ed., Oxford University Press, Oxford, (2018).

[8] Wang, Xiaomin, Fei Ma, and Bing Yao. ”Arbitrary degree distribution networks with perturbations.” AIP Advances 11.2 (2021): 025301.

[9] Tewari, Saurabh, and Leonard Kleinrock. ”Analysis of search and replication in unstructured peer-to-peer networks.” ACM SIGMETRICS Performance Evaluation Review 33.1 (2005): 404-405.

[10] Breslau, Lee, et al. ”Web caching and Zipf-like distributions: Evidence and implications.” IEEE INFOCOM’99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320). Vol. 1. IEEE, 1999.

[11] Kreitz, Gunnar, and Fredrik Niemela. ”Spotify–large scale, low latency, P2P music-on-demand streaming.” 2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P). IEEE, 2010.

[12] Deshpande, Varun, Hakim Badis, and Laurent George. ”Efficient topology control of blockchain peer to peer network based on SDN paradigm.” Peer-to-Peer Networking and Applications 15.1 (2022): 267-289.