Optimal Super-peer Selection for Large-scale P2P System

**Su-Hong Min, **Joanne Holliday, *Dong-Sub Cho*

**Computer Science and Engineering Department, Ewha Womans University*
shmin@ewhain.net, dscho@ewha.ac.kr

***Computer Engineering Department, Santa Clara University*
jholliday@scu.edu

Abstract

The peer-to-peer (P2P) systems have grown significantly over the last few years due to their potential for sharing various resources. Unstructured hybrid P2P system can improve the performance of the entire network and system using SP (Super-peer), which has the responsibility for query processing instead of OPs (Ordinary-Peer). In these systems, selecting the best SP to join is an important problem, but it is difficult to choose the optimal SP by the various reasons such as heterogeneous capacity, content similarity and dynamic capacity change. In this paper, we present the SP selection's problem and SP selection strategy based on dynamic capacity of the SP and content similarity. Also we measure the SP's score with weight to the factors distance cost, processing power and content similarity. Through the simulation, we show query processing performance is improved when OPs use our strategy to choose the best SP.

1. Introduction

Peer-to-Peer systems have recently received considerable attention from the networking research community. Initially, only pure P2P systems were considered. That is, all peers were assumed to have the same capabilities. Such systems use a flooding based search in which peers would send a lot of messages in their search to find the desired resource. However, peers with limited capabilities can cause bottlenecks. To improve this situation, a super-peer based hybrid P2P model was proposed. The hybrid P2P model which has been found very effective, is unstructured except for the division of peers into two layers, SP (Super-Peer) and OP (Ordinary-Peer). Each SP is connected to a set of Ops, while OPs are connected to only one SP. The SP deals with all queries instead of OPs [1, 2] so that an OP with low capacity is able to fully participate in the network. It is a feature of the Super-peer based P2P system that the OP should select only one SP for sharing resources and can participate in the network only through the chosen SP. Compared with pure P2P systems, Super-peer based P2P systems have to deal well with a large number of queries from OPs. The selected SP must handle queries efficiently and search for files requested by the OP. Existing systems break peers into OPs and SPs by considering only static capacities, such as the system specifications of peers. These systems pay little attention to a SP's dynamic capacity changes and characteristics of its resources. Often they use simple strategies such as random selection, when an OP chooses a SP. Although this technique is simple, it does not deal well with the heterogeneity of the participating peers both in terms of dynamic capabilities and a content similarity [15]. The SP selection problem is highly challenging because, in the P2P network, a large number of SP must be selected from a huge and dynamically changing network in which neither the peer's characteristics nor the network topology are known a prior [18]. Since the original purpose of super-peer based P2P systems was to improve performance, we should consider search efficiency and network performance in the design of the system. In this paper, we propose an SP selection strategy, to enable an OP to join the best SP considering a combination of dynamic capacity and similarity. First, we extract and evaluate three factors such as a distance cost, a processing power and a content similarity on the basis of dynamic capacities and the content similarity. We use simulation results to tell us the relative importance of these factors. Second, we compute a weighted score for the SP using these three factors so that OPs can rank the prospective SP.

The main contributions of this paper are: We present a method for SP to measure its distance, processing power and content similarity of the OPs in its cluster. OPs can consider capacity and similarity when they join a SP and the SP can use the content profile to improve searches. We demonstrate through simulation improved response time to file searches and reduced the consumption of bandwidth.

The rest of the paper is organized as follows: Section 2 reviews some related work briefly. Section 3 deals with our Super-peer selection strategy, it analyzes Super-peer selection problem and provides the weight based best super-peer by considering both capacity and similarity; Section 4 shows performance evaluation of proposed strategy. We conclude and propose the future work in Section 5.

2. Related Work

Recently hybrid P2P model based on Super-peers have supplanted the original pure P2P models. Super-peer based systems classify nodes into SP and OP by considering their capabilities. A super-peer is a node that acts as a centralized server to a subset of clients. These clients (OP) submit queries to their SP and receive results from it. In order to process the query for its OPs, each SP keeps an index of its OP's data. Yang and Garcia-Molina [1] examined the performance tradeoffs in super-peer systems. They and other researchers studied the potential drawbacks of super-peer network and reliability issues [1, 16]. Representative applications based on Super-peer architectures are Gnutella and KaZaA. In case of Gnutella, the original protocol, 0.4 version treats all nodes equal regardless of their bandwidth, CPU power etc. However, it has been observed that the abundance of signaling was a major threat to the scalability of these networks [2]. Gnutella 0.6 version suggested therefore the introduction of a two-level peer hierarchy: ultrapeers and leaf nodes [4]. SP and OP are named ultrapeer and leafnode in Gnutella. A principle goal of the Gnutella 0.6 architecture is to reduce the high message load, which can be observed in a Gnutella 0.4. The Gnutella specification [4] presented some criteria for ultrapeers such as not fire walled, suitable operating system, sufficient bandwidth and sufficient uptime. Peers classified as ultrapeers acts as proxies on behalf of less capable nodes, the so-called leaf nodes. Hence, when leaf nodes connect to one of ultrapeers, they don't consider physical capacities and user behavior between peers. Instead, they simply send Ping message including hop count to connect to ultrapeers in the network [12]. In case of KaZaA, although it is the most popular application based on super-peer architecture, it uses a proprietary protocol with encryption and little is known about the protocols, architectures, and behavior. A recent paper [3] presented the research community with an understanding of how KaZaA operates through some experiments. KaZaA was one of the first P2P systems to exploit this heterogeneity by organizing the peers into two classes, SN (Super Nodes) and ON (Ordinary Nodes) [13]. Like Gnutella, SNs are more powerful in terms of connectivity, bandwidth, CPU power, and non NATed accessibility. When an ON launches the KaZaA application, the ON chooses a parent SN, maintains a semi-permanent TCP connection with its parent SN, and uploads to this SN the metadata for the files it is sharing.

It has been shown that hybrid systems can reduce network traffic by using SP to handles queries of OP with low capabilities and to locate neighbors and resources. However, current approaches have only a simple strategy for selecting SP. A better way of evaluating potential SPs would further improve the network. We present an efficient SP selection strategy that considers physical capabilities and similarity service in hybrid P2P system.

3. Super-Peer Selection Strategy

In this section, we describe the Super-Peer selection strategy. Standard networking techniques such as random selection of peers, or flooding-based search for suitable SP have significant drawbacks. SPs selected by a random strategy may not be the best and flooding-based searches for a large number of SP neither scales nor adapts well to dynamic changes in network [18]. Therefore, when the OP selects the SP, we must consider how well the SP can deal with queries to provide OP's requested files as accurately and quickly as possible. First, we present the problem of SP selection and then we calculate the network distance between OP and SP using ASP, and measure dynamic capacity and content similarity in order to provide the optimal SP.

3.1 Super-Peer Selection Problem

In hybrid P2P system, an OP selects one SP to send queries and share resources. The OP is shielded from all other traffic and queries by other peers by its chosen SP. Since the OP depends on SP's capacities, the OP should select the SP which can provide it with the best service. It would be good if nearby SPs could provide the OP with a score so that the OP could choose the SP with the best score. The question is: how does the SP measure its own capacity so that the score is helpful to the OP in selecting the best SP? Available processing capacity (which is constantly changing) and content similarity are undoubtedly important, but how important are they to SP selection? The problems with SPs measuring their own capacities are as follows: Second, how should the SP measure its own physical capacities (available CPU, memory and disk I/O) when these vary dynamically according to the queries from cluster OP as well as SP's other work (generally, users use P2P systems with other work like word processor, e-mail, and Internet surfing etc. and would be annoyed if the P2P work caused perceivable slow-downs. Also, assuming the SP had a good measure of its capacities, which factor is the most important to the OP? For all of these reasons, it is difficult for SP to accurately evaluate its physical capacity at a given time.

3.2 Location based SP classification

To select the best SP, OP should choose the SP in the closest location. The primary goal of estimating network distance is to enable measurement of the network distance between arbitrary peers without direct measurement between SPs and OPs. If we directly calculate the distance of peers, it is too costly and time-consuming. To do this, we calculate the network distance between OP and SP using GNP algorithm. Several approaches have been proposed among which GNP may have received the most attention. GNP transforms the original distance data space into a Cartesian coordinate system and uses coordinates in the coordinate system to represent the location [6]. In this paper, we modify the existing GNP to adapt our system. We divide peers into three type, ASP (Agent Super-peer), SP (Super-peer), and OP (Ordinary-peer). ASP manages SP's join and leave and provides the b t SP to OP. Also, ASP operates as landmark. OP calculates the network distance between OP and ASP instead of measurement of distance between all SPs and OP for selecting the nearest SP. First, OP measures its RRT to ASPs and orders the ASPs in order of RTT. We classify the range of possible latency values into a number of levels. For example, we separate the range of possible latency values into 3 levels; level 0 (L_0) for latencies in the range [0,100]ms, level 1 (L_1) for latencies between [100,200]ms and level 2 (L_2) for latencies greater than 200ms. We then increase the ASP ordering of an OP with a level vector, one level number corresponding to each ASP in the ordering [19]. OP selects ASP with the best level vector, and requests SP's information to ASP. Second, the network distance between SP and ASP is calculated as the Euclidean distance in the Cartesian coordinates to provide the SP's distance information to OP. APS orders SPs and provide the best SP to OP. We divide the range of possible Euclidean distance values into 3 levels; level 0 (l_0) for distance in the range [0,2], level 1 (l_1) for distance between [2,4] and level 2 (l_2) for distance greater than 4. Through this procedure, we can calculate the network distance without direct measurement between OP and SP. Fig. 1 shows how new OP selects ASP and SP. To participate in the network, new OP should select ASP so that OP decides on an ASP by measurement of distance using RTT. In Fig. 1 OP selects ASP2 with (L_0). And then OP request SP's information to selected ASP. ASP can provide the optimal SP with (l_0) to new OP based on Euclidean distance value between SP and ASP.

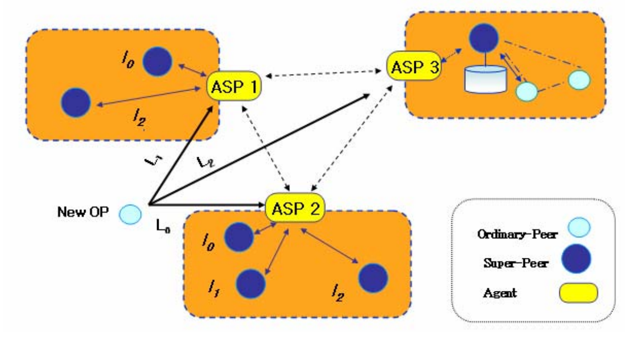

Fig. 1 Location based SP classification

3.3 Dynamic Capacity Measurement

A SP's processing power is a very significant factor in its ability to handle a large amount of network traffic and process the search queries for its OP's. Existing systems use stochastic techniques considering packet size or rely on SP's CPU specification as a measure of processing power. In this paper, we present three step measurement methods to provide more exact status information using a modified EMA (Exponential Moving Average). We observe that CPU usage changes dynamically with various conditions so that current CPU usage does not accurately describe the CPU's status and doesn't predict future CPU load when another new peer connects. In order to quantify CPU processing power, we modify EMA, a time series which gives more weight to more recent measurements than to other historical data. The original EMA is calculated using the previous EMA value and current CPU usage [7, 8]. We modified the EMA calculation for two reasons. First, the EMA value is very sensitive to the time interval used. If the time interval is too large, it does not reflect the recent CPU load. If the interval is too small, we have to continually calculate the EMA, causing a lot of overhead. Second, the existing EMA can accurately predict CPU load in a stable environment by observing CPU load changes for a few days. But P2P environment is very dynamic and thus pure EMA does not predict CPU load as well. Hence, in this paper, we modify EMA to respond quickly to new values at different time intervals.

To calculate dynamic processing power, we measure the CPU load in 3 steps. Each step has a different time interval. First, we calculate the current CPU usage using short intervals. Second, we calculate the Moving Average (MA) with the history dataset from step one using larger intervals. Finally, we measure CPU load with modified EMA using the largest time interval.

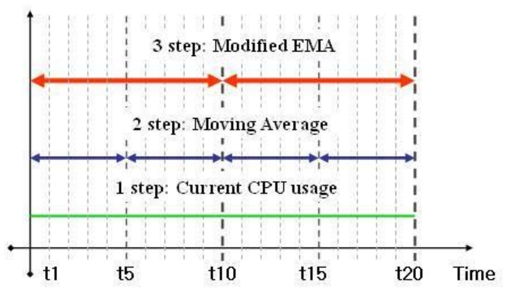

Fig. 2 Step by step CPU load measurement

Fig. 2 shows the three different steps and their time intervals. In this example, the whole period is 20 minutes, current CPU usage, C_t is measured a total of 20 times (once per one minute), MA_t is calculated 4 times at 5 minute intervals, and EMA_t is calculated twice at 10 minute intervals. The CPU load of SP, C_μ , is evaluated using the previous EMA and MA.

The formula for cost of the processing power is

$$C_t = \{c_1, c_2, c_3, \ldots\ldots, c_N\} \quad (1)$$

$$MA_t = \frac{\sum_{i=0}^{n-1} C_{t-i}}{n} \quad (2)$$

$$C_\mu = \alpha \cdot MA_t + (1-\alpha) \cdot EMA_{t-l} \quad (3)$$

C_t is a set of current CPU values at t time intervals. *MA* is an average value of C_t, and MA_t is a set of *MA* values, $MA_t = \{ma_1, ma_2, \ldots, ma_t\}$. C_μ is current *EMA*, and it is the predicted CPU load. EMA_{t-l} is previous exponential moving average. *N* means the number of periods for *EMA*. α is a smoothing constant, α= 2/(N+1). The smoothing constant applies the appropriate weighting to the most recent value relative to the previous *EMA*. In this way, we get a precise measure of processing power that helps OP to select the best SP.

3.4 Content based Similarity Measurement

We consider a content similarity which can help OP to obtain the most relevant content and files in a very short time. OPs should be able to choose a SP which has other OPs in its cluster with similar content. To do this, OPs should decide on content categories and then pick suitable SP to the category of interest. Second, the SP should make a content model from the information learned through the queries of its OPs and also maintain a profile on the OPs in its cluster according to the categories. To do this, we use ISM algorithm which uses Nearest Neighbor classification technique and cosine similarity to calculate the similarity value of content respectively. It consists of two components: a profile mechanism and a relevance rank. The SP computes each peer's content similarity and ranking with them. The profile mechanism is that SP builds a profile for each of its OPs. The profile keeps the most recent replies of each peer. The RelevanceRank, which is a cluster ranking mechanism uses the SP's profile to select the best clusters for answering a query [10]. The RelevanceRank (RR) is calculated as follows:

$$RR_{Pl}(P_i, q) = \sum Qsim\ (q_j, q)^\alpha \times S(P_i, q_j) \quad (1)$$

$$Qsim\ (q_j, q) = \frac{q_j \cdot q}{|\overrightarrow{q_j}| \times |\overrightarrow{q}|} = \frac{\sum_{t=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} W^2_{i,j}} \times \sqrt{\sum_{t=1}^{t} W^2_{i,q}}} \quad (2)$$

SP(c) = (QueryHit / TotalHit) + (peers / totalPeers) (3)

The formula (2) is the cosine similarity and $S(P_i, q_j)$ is the number of results returned by Pi for query qj. The formula (1) allows us to rank higher the clusters that returned more results. In addition, the parameter, α, allows us to add some weight for the most similar queries. The formula (3) is a content similarity's score of SP, it calculates the number of query hits returned by clusters to total query hits and the number of cluster returned results to the total number of peers, that is, results from other SP's clusters.

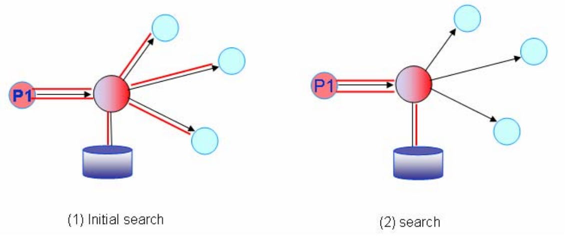

Fig. 3 Content Similarity based Information Retrieval

In order to process the query for SP's cluster, the SP keeps the profile which includes query hit information. First, if the SP receives a query from the OP to search files, the SP transmits it to its cluster by broadcast as Fig 3. If the SP finds any results, it will return one response message, and update the profile with the query hit. After awhile, the SP builds up an accurate profile. This helps SPs respond quicker and more precisely to OPs.

3.5 Weight based Super-Peer Selection

There are many metrics that may be used to select the best SP, such as average response time, bandwidth, and so on. These metrics may have different weights depending on the objective. In case of network administrators, their concern is how much network bandwidth is consumed by a P2P system, while an OP cares more about the response time of queries. In this section, we focus on the OP's preference so that we compute SP's score to predict the average response time when the OP searches files using the SP. In order to make a ranked set of SPs, OP must obtain scores of all SP's in the limited range such as a given hop count. We compute the SP's score with weight to the factors distance cost, processing power and content similarity. The best weight value is determined by simulation in Section 4.

Through simulation in section 4, we experiment with different weights for these factors and measure total response time to queries from the OP. In this way, we can discover the optimal weight of the three factors. Among them, processing power most affects the SP's response time so it has the greatest weight value. It is followed by content similarity with a smaller weight and distance cost with the smallest weight value. The SP's score can be computed as,

$$SP_i = \alpha \times d_i + \beta \times p_i + \delta \times c_i \qquad (1)$$

d, p, and c show distance cost, processing power, and content similarity, respectively. α is the weight of distance cost, β is the weight of processing power And δ is the weight of content similarity. In this paper, the weight value can be calculated as following: β=0.5, (α+δ) <= 0.5 ($\alpha < \delta$).

4. Experiment Evaluations

In this section, we present the simulation model used to evaluate the performance of our SP selection method and determine the weights of the SP factors. We also discuss simulation results. The simulation model is implemented in C++ using CSIM [17]. The network setup and performance metrics are shown in Table 1. It consists of a number of OPs, SPs and ASPs. OPs join the network and request services or provide them during their lifetimes and then disconnect from the network. All peers repeat these processes during the simulation time.

Table 1 Default parameter settings

Parameters	Default Values
SIMTIME	100000
The number of OP	100 ~ 10000
The number of SP	10 ~ 100
The number of ASP	5 ~ 10
The range of distance	0 ~ 200ms
contents	10
CPU power factor	{1.0,1.5,2.0,2.5,3.0}
Max hop	7

In our simulation, we verify that the SP selection strategy can improve scalability and bandwidth cost by taking dynamic capacity and similarity into consideration. We evaluate it under various simulation environments. Figure 4 tells us that the whole performance could change depending on the priority of factors. We measure three factors for different weight values changes when the number of OP and SP is 1000 and 50, respectively. Figure 4 shows that among the three factors, processing power most significantly affects the SP's response time so it is given the greatest weight value, which is followed by content similarity and distance cost. In Figure 5 we perform an experiment measuring the average message response time for the different factors and for the optimal combination, "best SP". This experiment demonstrates the good performance of our selection of the best SP.

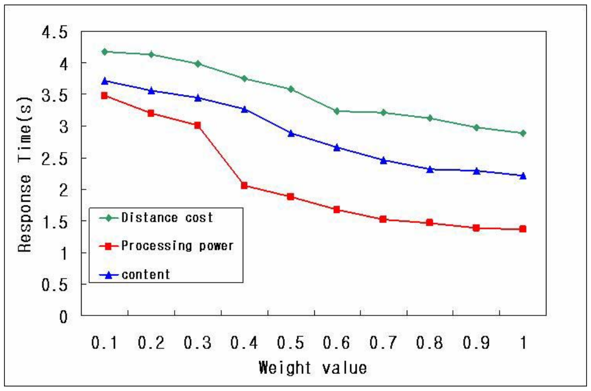

Fig. 4 Weight value vs. Response time

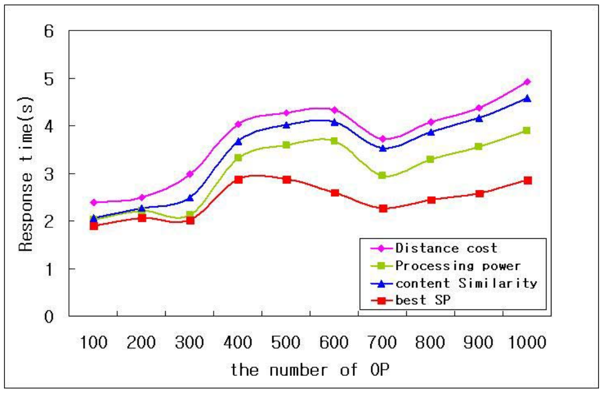

Fig. 5 Three factor vs. Best SP

5. Conclusion and Future Work

We have presented the Super-peer selection strategy for providing the best super-peer. In the Super-Peer based P2P systems, SPs are supposed to handle all queries received from OPs. As a result, the selecting SP is important problem, but it has received little attention from the research community and most of the existing systems only select a SP at random. In this study, we present the SP selection problem and SP selection strategy by analyzing capacity and similarity between peers. Also we compute the SP's score with weight to the three factors such as distance cost, processing power and content similarity. Through the simulation, we show the performance of the response time using the best SP. We plan to implement additional features in the future such as mechanisms that allows monitoring of traffic between peers and strategies for topology self-organization.

6. References

[1]B. Yang, H. Garcia-Molina, "Designing a super-pee network", IEEE International Conference on Data Engineering, Bangalore, India, Mar. 2003.

[2] Y. Chawathe , S. Ratnasamy , L. Breslau , N. Lanham and S. Shenker, "Making Gnutella-like P2P systems scalable", Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, Karlsruhe, Germany, Aug. 2003.

[3] J. Liang, R. Kumar, K Ross, "The KaZaA Overlay: A Measurement Study", Proceedings of the Fifth New York Metro Area Networking Workshop, Sep. 2005.

[4] Gnutella protocol spec. v.0.6 http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

[5] Suhong. Min, D. Cho, "An Intelligent Performance based Hybrid P2P System", Journal of Korea Electrical Engineering and Technology", 5(2). Feb. 2006

[6] T.E.Ng, H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approach", in INFOCOM, 2002.

[7] Box, G.E., Jenkins, G.M, "Time Series Analysis Forecasting and Control, Holden day, 1976.

[8] L. Yang, I. Foster, J. M. Schopf, "Homeostatic and Tendency-based CPU Load Predictions", IEEE International Parallel and Distributed Processing Symposium, 2003.

[9] Bo Ling , Zhiguo Lu , Wee Siong Ng , Beng Chin Ooi , Kian-Lee Tan , Aoying Zhou, "A Content-Based Resource Location Mechanism in PeerIS", Proceedings of the 3rd International Conference on Web Information Systems Engineering, Dec. 2002.

[10] V. Kalogeraki, D. Gnuopulos and D. Zeinalipour-Yazti, "A Local Search Mechanism for Peer-to-Peer Networks", Proceedings of CIKM'02, McLean VA, USA, Feb. 2002.

[11] D. Tsoumakos, N. Roussopoulos, "A comparison of peer-to-peer search methods", In proceedings of the sixth International Workshop on the Web and Database, San Diego, CA, June. 2003.

[12] Gnutella website: http://www.gnutella.com.

[13] KaZaA website: http://www.kazaa.com.

[14] Pslist website: http://www.sysinternals.com/Utilities/PsList.html.

[15] R. Steinmetz, Klaus Wehrle , "Peer-to-Peer Systems and Applications", Lecture Notes in Computer Science, Vol. 3485 Springer 2005.

[16] Li Xiao, Z. Zhuang, Y. Liu, "Dynamic Layer Management in Superpeer Architectures", IEEE Transactions on parallel and distributed systems, 16(11), Nov. 2005.

[17] CSIM Development toolkit for simulation and modeling. http://www.mesquite.com.

[18] V. Lo, D. Y, C. G, J.L, "Scalable Supernode Selection in Peer-to-Peer Overlay Networks", Proceedings of the 2005 Second International Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P'05), OR, USA, July, 2005

[19] S. Ratnasamy, M. Handley, R. Karp, S. Shenker, "Topologically-aware overlay construction and Server Selection", In INFOCOM'02, New York, 2002.