# A Control Theoretic Approach to Analyzing Peer-to-Peer Searching

Gang Ding
*Qualcomm Research*

## Abstract

A generic linear mathematical model is proposed to represent the dynamics of bandwidth usage, network topology, and host processing power in large scale Peer-to-Peer (P2P) networks. Feedback control theory is employed to analyze system stability and query controllability, as well as deriving an explicit solution for the bandwidth usage. The proposed model and analysis methods can be applied to various P2P networks, such as broadcast based P2P network, super-peer based unstructured P2P network, and distributed hash table based structured P2P network. The synthesis of the model is also presented, which adaptively adjusts query rate in order to properly control the bandwidth usage.

## 1. Introduction

Peer-to-Peer (P2P) overlay networks and applications built upon them have greatly prospered in the past decade. P2P networks enable information sharing among large number of users without significant assistance from dedicated servers. How to efficiently find target information is one of the major research topics on P2P networks. Napster and BitTorrent P2P networks indeed depend on centralized directory servers which maintain index information collected from publishers and peers. A peer should direct its query to these servers before downloading real data from other peers. Gnutella [1] first introduced an open and pure P2P network where each peer broadcasts its query to neighboring peers. And neighbors forward the query further until a target peer is found. But this type of query '*flood*' may take too much network bandwidth so that the low bandwidth peers become bottleneck. To address this issue, updated Gnutella [2] and other popular P2P networks such as KaZaA [3] adopted a super-peer concept, where some well-connected peers act as super-peers that receive queries from other low-bandwidth peers (called *leaf*) and search on behalf of its leaves. As a result, queries are only broadcast among super-peers. All these P2P searching methods are called *unstructured* because of the broadcast of queries. Some highly structured and efficient P2P searching algorithms have recently been proposed, for example, Chord [4] and Kademlia [5]. They all employed Distributed Hash Table (DHT) technique to share the searching control information among peers themselves. Most popular P2P networks have also integrated DHT-based search protocols.

While various searching algorithms have been proposed and implemented in large scale P2P networks, most performance analysis is based on intuition, observation, or limited measurement. This paper proposes a general mathematical model which can be employed to analyzing various P2P searching algorithms. Instead of static study, the proposed model captures the dynamics of system states. It can be formally analyzed by feedback control theory in state space. The synthesis of the model derives an adaptive algorithm that further improves or optimizes the system performance. The remaining of the paper is organized as follows. Next section briefly introduces related work. Section 3, 4, and 5 present the model, analysis, and synthesis, respectively. Last section concludes the paper with summary and further research directions.

## 2. Related Work

There are many varieties of unstructured [6] and structured P2P search algorithms [7]. A good review of them can be found in [8]. Some early performance analysis was based on measurements. [9] studied the population of peers participating Napster and Gnutella based on measurement in about a month. The peer behaviors, such as how often peers connect and disconnect from the system and the degree of cooperation between peers, are characterized. [10] employed light-weight measurement based techniques to optimize the peer selection performance. These studies only investigated the static behaviors in P2P networks. In [6], a probabilistic model was used to analyze the dynamic query behavior. [11] introduced Search/Index Link (SIL) as a graph representation of broadcast based P2P search network. [7] analyzed the resilience of structured P2P systems by

Markov-chain where the query might be wrongly forwarded with certain probability. This approach only works for the resilience analysis, with assumptions that all peers are evenly distributed. [12] provided a generic closed queuing network model for P2P file sharing systems. But the model did not help to analyze the searching capacity and other system criteria. [13] presented a simple fluid model and a game-theoretic model for BitTorrent-like networks. But the study is focused on the steady-state performance of data sharing. [14] tried to reverse engineer the Skype protocol by analyzing the Skype network traffic. But there is no real performance analysis involved. [15] proposed P2P searching algorithms for mobile wireless networks, but no analysis result is provided. [16] tried to optimize the random walk searching algorithm in unstructured P2P network. [17] is the first to introduce feedback concept to P2P systems in order to make it adaptable to the dynamics in a P2P system. But how to design the feedback system is guided by heuristics instead of control theory. In the following sections of this paper, a dynamic state model for P2P searching will be presented, analyzed, and synthesized via feedback control theory.

## 3. Model

The major resources of a P2P network are the communication bandwidth and peer's processing power. Designing a P2P search algorithm is affected by the following aspects of the system [18]:

- Data and/or index location: how far the target data itself or the index information of the target data is located from the requesting peer.
- Network topology: through which connection(s) a query can be sent out.
- Routing policy: broadcast based unstructured routing or structured routing.
- Processing power of peers: the speed of each peer to process incoming queries.

To accommodate them, a generic model is proposed:

$$\dot{x}_i(t) = \sum_{j=i}^{n} a_{ij} x_j(t) + \sum_{k=i}^{n} b_{ik} u_k(t), \; (i = 1, \cdots, n) \quad (1)$$
$$x(0) = 0$$

where $x(t) = [x_1(t), \;\; \cdots, \;\; x_n(t)]^T$ is the state vector of a system of $n$ peers. The state $x_i(t)$ of each peer $i$ represents the aggregated query data processed by this peer. Therefore, $\dot{x}(t)$ is the bandwidth used for searching.

The input vector $u(t) = [u_1(t), \;\; \cdots, \;\; u_n(t)]^T$ is the search rate of all peers. For the sake of simplicity, we assume that the search rate has already been converted to the corresponding query data rate with unit bits/second.

The state matrix $A(t)$ is affected by network topology, routing policy, and the processing power of each peer. Specifically, $a_{ii}(t)$ ( $i = 1, \cdots, n$ ) is negative, whose absolute value represents the processing rate of peer $i$. $a_{ij}(t)$ ( $i, j = 1, \cdots, n; j \neq$ i) are non-negative numbers denoting the rate of query data $x_j(t)$ from peer $j$ to peer $i$. $a_{ij}(t)$ is positive only if there is a communication link from peer $j$ to peer $i$ and peer $i$ is chosen as peer $j$'s next hop by the routing policy. In order to reduce redundant query forwarding, after a peer $i$ sends a query out, it will never process the same query again.

The matrix $B(t)$ represents how each query is sent out. If $b_{ik}(t)$ ( $i, k = 1, \cdots, n$ ) is positive, it means that the query originated from peer $k$ is first directed to peer $i$.

Basically, the general state space model [1] of a P2P system of $n$ peers involves all the aggregated query data states. The dynamic of each state $x_i(t)$ of peer $i$ is determined by the processing speed $a_{ii}(t)$ of peer $i$ itself and all the queries directed to it from other peers, where queries may be forwarded from another peer $j$ through link $a_{ij}(t)$, or directly from another peer $k$ with query rate $u_k(t)$ and coefficient $b_{ik}$. The reason that $a_{ii}(t)$ is negative is that each peer processes incoming queries and always keeps making the incoming data smaller by a negative rate.

## 4. Analysis

Given the general model (1), we will analyze it by answering following fundamental questions.

*1) Is the whole system stable? That is, is it possible that a peer's state keeps increasing to infinity or oscillating without converging asymptotically?*

***The answer is that the whole system is stable and will go arbitrarily close to the static state after finite period of time.***

The proof is as follows. First of all, due to linear nature of the state space model (1), we can investigate every query input $u_i$ separately so that the actual state is the summation of the states corresponding to each input $u_i$. Second, without loss of generality, we assume a single query input $u_1$, then we can switch the position of every pair of states $x_i$ and $x_j$ when $i < j$ and $a_{ij} > 0$. By this way, the peer closer to the origin of query will have smaller index in the transformed state space model below,

$$\dot{\hat{x}}(t) = \hat{A}(t)\hat{x}(t) + \hat{B}(t)u_1 \qquad (2)$$
$$\hat{x}(0) = 0$$

where $\hat{x}(t)$ is a re-ordered state vector of $x(t)$ and the new state matrix $\hat{A}(t)$ is of a lower triangular form:

$$\hat{A}(t) = \begin{bmatrix} \hat{a}_{11} & 0 & \cdots & 0 \\ \hat{a}_{21} & \hat{a}_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ \hat{a}_{n1} & \hat{a}_{n2} & \cdots & \hat{a}_{nn} \end{bmatrix}_{n\times n}, \qquad \hat{B} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n\times 1}$$

This is due to the fact that after switching positions, states with lower index will not accept the same redundant queries from higher index states. Also note that this kind of orthogonal transformation does not change the eigenvalues of the state matrix $A(t)$, neither does the stability property. Therefore, we just need to investigate the stability property based on the new state matrix $\hat{A}(t)$ which is easily proven to have eigenvalues: $a_{11}$, …, $a_{nn}$, which are all negative. This implies that all states are asymptotically stable [19].

*2) How far can a query go, or can a query reach all other peers in a given network topology?*

***The answer is that it is not always true that a query can reach all peers. It is true only when the following condition holds:***

$$\hat{a}_{i(i-1)} \neq 0, \text{ for } i = 2, \ldots, n \qquad (3)$$

This condition implies that, in order to propagate a query to all peers, every peer should have a network connection to its immediate next neighbor in state vector $\hat{x}(t)$. The proof is based on the definition of *controllability*: a linear system is controllable if there exists an input $u(t)$ such as to drive the state $x(t_0)$ at time $t_0$ to an arbitrary value $x(t_f)$ in a finite period of time $t_f - t_0 \geq 0$. For the general model (1), the controllability property is equivalent to the following condition [19]:

$$rank[A - \lambda_i I \quad B] = n \qquad (4)$$

where $\lambda_i$ is one of the $n$ eigenvalues of $A$. By this condition and the fact that orthogonal transformation does not change controllability, and the eigenvalues of $\hat{A}(t)$ are $a_{11}$, …, $a_{nn}$, the sufficient condition for a query $u_1$ to be propagated to all peers is (3). If condition (4) is not met, it is straightforward to find how far a query can go by looking for the first zero from sequence $\hat{a}_{i(i-1)}$ ($i = 2, \ldots, n$).

*3) Is there a closed form solution to all states in model (1) and (2)?*

***The answer is yes and the solution is:***

$$\hat{x}_1(t) = (u_1/-\hat{a}_{11})(1 - e^{\hat{a}_{11}t}) \qquad (5)$$

$$\hat{x}_k(t) = L^{-1}\{\sum_{i=1}^{k-1} \frac{\hat{a}_{ki}\hat{x}_i(s)}{(s - \hat{a}_{kk})}\}, \qquad (k = 2, \ldots, n)$$

where $L^{-1}\{.\}$ represents the inverse Laplace transform of {.}, and $x(s)$ is the Laplace function of $x(t)$.

Following the same argument as that in question 1), we only need to find a solution for model (2) with a single input $u_1$. Due to the particular form of $\hat{A}(t)$, the above solution can be found step by step; every state $\hat{x}_k(t)$ is derived from solutions for previous states. Noticing that in frequency domain, state $\hat{x}_k(s)$ is the result of adding a pole $\hat{a}_{kk}$ and a productive constant $\hat{a}_{ki}$ to every previous state $\hat{x}_i(s)$. So in time domain, an extra decreasing exponential $e^{a_{kk}t}$ is added to the dynamic response.

Approximation can be done to simplify solution (5) and give more insights of the state dynamics. Below we demonstrate this in three special cases.

**Case 1)** If eigenvalues $a_{ii}$'s are significantly apart from each other, then the eigenvalue with the smallest absolute value, say $a_{mm}$ will dominate the final result because its corresponding exponential $e^{a_{mm}t}$ changes the slowest. Hence the time spent to propagate a query is controlled by the bottleneck peer $m$ that has the lowest processing power. According to (5), the influence of peer $m$ will be propagated to all its downstream peers.

**Case 2)** In addition to finding the propagation delay, the dynamic solution (5) can be used to determine how the distribution of eigenvalues affects the states. For example, given two different eigenvalues $a_1$ and $a_2$, the corresponding time domain state includes $\left|e^{a_1 t} - e^{a_2 t}\right| / \left|a_1 - a_2\right|$, which achieves the maximum at time $t_{\max} = \left|\ln|a_1| - \ln|a_2|\right| / \left|a_1 - a_2\right|$. This clearly shows that the larger the difference between two eigenvalues, the faster the state reaches the maximum, with higher potential of overloading the corresponding peer.

**Case 3)** If there are $r$ peers having similar processing rate $a$, then the solution of every following peer also includes the dynamics with Laplace representation: $\sum_{i=1}^{n} 1/(s-a)^i$. The corresponding time domain expression is $\sum_{i=0}^{k-1} e^{at}t^i/i!$, which is still dominated by the processing rate $a$, although the time delay becomes a little longer due to $t^i$'s. For each element in the above state dynamics, the maximal data throughput is achieved at time $t_{i\max} = -i/a$, which is inversely proportional to the processing rate $a$.

Above approximations give some insights of the dynamics of aggregated query and query propagation rate:

- When peers are of different processing power, the slowest peer will dominate the query propagation rate. And the larger the difference between peers, the faster the aggregated query will saturate.
- When the peers are of similar processing powers, their common processing rate will dominate the query propagation rate, and the faster the processing rate, the sooner the aggregated query will saturate.
- The above observations suggest a possible performance improvement by adaptively updating the state matrix $A(t)$ and query input matrix $B(t)$ so that less query traffic will be directed to the low-capacity peers. This requires the knowledge of processing rate of other peers, which could be obtained by exchanging such information between neighbor peers.

Given the general model (1) and analysis, below we will demonstrate how to apply them to some existing P2P searching algorithms.

### 4.1. Pure Broadcast Search

Old version of Gnutella employs pure P2P query broadcast to all neighbors, which corresponds to the model (1) where every element $a_{ij}$ in $A(t)$ is at its maximal value as far as there is a communication link between peer $j$ and $i$, because every peer is supposed to forward all received queries to all its neighbors. Matrix $B(t)$ is simply an identity matrix where diagonal elements are 1's and all the other elements are 0's, because every requesting peer is supposed to initiate a query by itself.

In addition to the above generic analysis, we can derive following characteristics specific to pure broadcast search based P2P networks. Since $B(t)$ is an identify matrix of full rank, the controllability of overall system is guaranteed by (4). This corresponds to the special case that at least every peer can generate arbitrary queries by itself which makes its own state controllable. But in order for a particular query to reach any other peers in the network, we still need (3) to be held, which has the highest probability to be true in pure broadcast search P2P networks because all available communication links are used for query forwarding.

Due to the full use of communication links in matrix $A(t)$, the state dynamics have their most complicated form in (5), which makes queries most likely to be delayed by low-capacity peers. And the state has the highest probability to quickly reach the maximum.

### 4.2. Broadcast Search with Super-peer

In this kind of P2P networks, a small set of super-peers are chosen to take responsibility of forwarding queries for all peers in the system. Other peers only need to send their indexes and queries to, as well as receive search results from, its corresponding super-peer. To model a super-peer based P2P network, we assume there are $n_0$ ($< n$) peers chosen as super-peers, and each super-peer $i$ is responsible to a set of normal peers, named *Leaf*($i$). In model (1), for every super-peer $i$, $a_{ij}$ = 0 and $b_{ij}$ = 1 for any $j \in Leaf(i)$. This means that queries from leaves are only addressed to their super-peers. Therefore, we only need to model the dynamic states of super-peers. We here ignore the data traffic of sending indexes, queries and search results between a super-peer and its leaves because they are relatively smaller than query traffic among super-peers. The reduced model is:

$$\dot{x}_s(t) = A_s(t)x_s(t) + B_s(t)u_s(t) \qquad (6)$$

$$x_s(0) = 0$$

where state vector $x_s(t)$ is of dimension $n_0$ and each input query $u_{si}, (i = 1, \ldots, n_0)$ is the summation of all queries from $i$th super-peer's leaves:

$$u_{si} = \sum_{j \in Leaf(i)} u_j$$

The controllability condition of the reduced model is similar to (3), which implies that the selected super-peers should be strongly connected to each other such that every query can reach any other super-peers.

The solution to the reduced model (6) can be similarly calculated by (5). So the previous generic analysis still applies to the reduced model of super-peers. It is obvious that the query propagation rate is limited by the super-peer of the lowest capacity. This means that every super-peer should be carefully selected so that no low-capacity peers are chosen – even one low-capacity peer can significantly deteriorate the performance of the whole system.

Intuitively, it is expected that the adoption of super-peers will greatly improve the successful search rate – hopefully proportionally to the number of leaves of a super-peer. But the above analysis does not support this due to a couple of reasons. First, the query propagation rate, which is approximately proportional to the successful search rate, is mainly determined by the processing speed of super-peers. Hence the performance improvement is determined by the ratio of processing speed between super-peers and its leaves, not the ratio between the number of total leaves in the system and super-peers. Second, according to (5) and noticing the linear nature of the model, when the input query of a single peer, $u_1$, in the general model (1) is replaced by the aggregated queries from all leaves of a super-peer, $u_{s1}$, in the specific model (6), the aggregated query data $x(t)$ is also increased by a constant $u_{s1}/u_1$. When the aggregated query exceeds the available bandwidth of a super-peer, it will be overloaded and the system performance could even be worse in this case.

The above analysis shows that the performance improvement of super-peer based P2P networks highly depends on the careful selection of super-peers. The following selection policy should be followed:

- In order to increase the successful search rate by $c$ times, the lowest processing rate of all super-peers should be about $c$ times faster than the lowest processing rate of all leaves.
- In order to avoid overloading super-peers, the assigned bandwidth of a super-peer should be much larger than a regular peer. It is preferred to provide at least the summation of required bandwidths for all its leaves.

### 4.3. DHT-based Search

DHT-based searching algorithms are using specific routing table to forward queries in O(log$n$) hops, which makes the searching process much more efficient than broadcast based approaches. Using the general model (1), for a particular input query for key $k$, after looking up the routing table of size O(log$n$), it will be forwarded to a single peer, and so on. If the query traverses $m$ peers, then the data throughput for each intermediate peer is derived from (5) as:

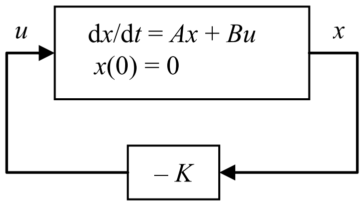

Fig. 1. System with feedback controller

$$x_k(t) = L^{-1}\{\frac{u_1}{s}\prod_{j=1}^{k-1}\frac{1}{(s-a_{jj})}\}, \qquad (k=1,\ldots,m)$$

Since there are only $m$ = O(log$n$) peers out of n peers being affected by a query, the cost due to searching becomes less important. On the other hand, there are other control overhead associated with a DHT based searching algorithm. Further investigation is beyond the scope of this paper.

## 5. Synthesis

Above we have investigated some existing P2P networks using proposed model and system analysis theory. It is shown that every approach has its own advantages and disadvantages. From the feedback control point of view, however, modeling and analyzing of a system is not enough if we want to achieve better system performance. The synthesis of a system is employed to design a feedback controller so that the system output will track what we desire automatically. Looking at the general model (1), we notice that the input query rate $u(t)$ has been assumed to be any value chosen by the user. To synthesize the model, we will specify $u(t)$ by the form

$$u(t) = -K\,x(t), \tag{7}$$

which means that the input query rate $u(t)$ will be adapted according to the actual system output $x(t)$ (see Fig. 1). Actually, we can control all the dynamics of state $x(t)$ by choosing an appropriate feedback controller $K$.

We here use Linear Quadratic Controller (LQC) design method to synthesize model (1). For the sake of simplicity, we assume that $\hat{A}(t)$ and $\hat{B}(t)$ are time invariant, which is reasonable because even when an element is varying, its update rate is much slower than that of state $x(t)$ so that it can be regarded as constant. Our objective is to find a $K$ such that the quadratic performance

$$J = \int_o^\infty [x^T(t)Qx(t) + u^T(t)Ru(t)]dt \tag{8}$$

is minimized, where $Q = M^T M$ is positive semi-definite, $R$ is positive definite. $Q$ and $R$ are state and control weighting matrices, respectively.

It is well known [19] that if both $(A, B)$ and $(A, M^T)$ are controllable, there exists a unique optimal controller $u^0(t)$ which minimizes the above quadratic performance $J$, and the optimal $K$ is

$$K = R^{-1}B^T P^0$$

where $P^0$ is the unique positive definite solution of the matrix algebraic Riccati equation:

$$A^T P + PA + Q - PBR^{-1}B^T P = 0 \qquad (9)$$

Given $A$, $B$, $Q$ and $R$, however, there is no closed form solution for $P^0$. But there are many efficient numerical computing methods, such as Schur's algorithm [20], to quickly find $P^0$.

In order to apply the above controller synthesis method to a real P2P network of $n$ peers, however, the dimension $n$ of the above matrices might be too large to calculate $P^0$. In addition, even if $P^0$ can be found, when we apply the feedback controller (7) to the real system, each peer has to know the current state $x(t)$ for every peer in the system, which is not feasible. Therefore, we will apply the above synthesis method in a distributed way: for each peer, we do the following steps:

- First construct a small local system model involving the states of the peer itself and its neighbors;
- The peer communicates with its neighbors periodically in order to get correct $A$, $B$, and $x(t)$;
- Numerical computing method is used to find $P^0$;
- $K$ is further calculated from $P^0$;
- Finally according to (7), the control input $u(t)$ is calculated from $K$ and state $x(t)$.

This distributed synthesis protocol may not be able to minimize the global performance $J$, but it at least achieves locally optimized performance.

This synthesis algorithm implies that the user query rate $u(t)$ is actually not determined by users. Instead, it is dynamically adapted based on the available communication resources and the states of neighbors. When users generate queries faster than the calculated $u(t)$, the client software should hold some queries until the synthesized $u(t)$ becomes big enough.

The above synthesis algorithm aims at minimizing the performance criterion (8) which tries to obtain the smallest aggregated query state. In reality, however, smaller aggregated query may limit the search range and the number of successful search results. A better approach is that under the constraint of maximal available bandwidth, we can assign the desired bandwidth, denoted as $x_{ref}$, which should be consumed by peers in order to get satisfactory searching results. Given $x_{ref}$, the above synthesis can be slightly modified to deal with the difference between the real state $x(t)$ and reference state $x_{ref}$. Define:

$$e = x_{ref} - x.$$

Then the new quadratic performance is defined as

$$J_t = \int_o^\infty [e^T(t)Qe(t) + u^T(t)Ru(t)]dt\,.$$

Under the same conditions, the new control input $u$ can be found as:

$$u = -R^{-1}B(P^0 x - \mu)\,, \qquad (10)$$

where $P^0$ is the solution of (9), and $\mu$ is calculated by

$$\mu = [(A - BR^{-1}B^T P)^T]^{-1} Q x_{ref} \qquad (11)$$

## 6. Summary

This paper presents a general mathematical model for the dynamic behaviors of query during P2P searching. The model is a state space representation involving the aggregated query, network topology and bandwidth, and host processing power. Modern control theory is employed to analyze the stability, controllability, and state dynamics. The range of queries is analyzed by controllability, while the query propagation rate is determined by the state dynamics which can be explicitly solved from the model. The proposed model and analysis methods are applied to existing P2P networks. The synthesis of the proposed model is also presented based on optimal feedback control theory, which adaptively adjusts peer's query rate in order to track desired network bandwidth usage automatically.

This is one of the first attempts to model, analyze, and control P2P searching using feedback control theory. It builds a generic theoretical framework for analyzing the dynamic behaviors of current and future P2P searching algorithms. Future study will be focused on applying feedback control theory to model, analyze, and synthesize P2P data downloading and the combined performance due to both searching and downloading, as well studying the performance of P2P networks in mobile wireless environment.

## 7. References

[1] S. Osokine, "The Flow Control Algorithm for the Distributed 'Broadcast-Route' Networks with Reliable Transport Links," http://www.grouter.net/gnutella, Janurary, 2001.

[2] S. Osokine, "Search optimization in the distributed networks," http://www.grouter.net/gnutella, December 2002.

[3] J. Liang, R. Kumar, and K. Ross, "The KaZaA Overlay: A Measurement Study," Computer Networks, 50:842-858, April 2006.

[4] I. Stoica, *et al.*, "Chord: a scalable P2P Lookup Protocol for Internet Applications," SIGCOMM, 2001.

[5] P. Maymounkov and D. Mazières, "Kademlia: A P2P Information System Based on the XOR Metric," The 1st International Workshop on Peer-to-Peer Systems, 2002.

[6] B. Yang and H. Garcia-Molina, "Comparing Hybrid P2P Systems," VLDB, 2001.

[7] S. Wang, D. Xuan, and Wei Zhao, "On Resilience of Structured P2P systems," IEEE Globecom, 2003.

[8] C. Kang, "Survey of Search and Optimization of P2P Networks," Peer to Peer Networking and Application, 4(3), 2011.

[9] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A Measurement Study of P2P File Sharing Systems," Multimedia Computing and Networking, 2002.

[10] T. Ng, *et al.*, "Measurement-based Optimization Techniques for Bandwidth-demanding P2P Systems," INFOCOM, 2003.

[11] B. Cooper and H. Garcia-Molina, "SIL: Modeling and Measuring Scalable P2P Search Networks," The International Workshop on Databases, Information Systems and P2P Computing, 2003.

[12] Z. Ge, D. Figueiredo, S. Jaiswal, J. F. Kurose, and D. Towsley, "Modeling Peer-Peer File Sharing Systems," INFOCOM, 2003.

[13] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-like P2P Networks," SIGCOMM, 2004.

[14] S. Baset and H. Schulzrinne, "An Analysis of the Skype P2P Internet Telephony Protocol," The 25th INFOCOM, 2006.

[15] G. Ding and B. Bhargava, "Peer-to-Peer File-sharing over Mobile Ad Hoc Networks," The 1st International Workshop on Mobile P2P Computing, 2004.

[16] N. Bisnik and A. Abouzeid, "Optimizing Random Walk Search Algorithms in P2P networks," Computer Networks, 51(6): 1499-1514, 2007.

[17] E. Khatibi, *et al.*, "Dynamic Multilevel Feedback-Based Searching Strategy in Unstructured Peer-to-Peer Systems," IEEE International Conference on Green Computing and Communications, 2012.

[18] N. Daswani, H. Garcia-Molina, and B. Yang, "Open problems in data-sharing peer-to-peer systems," ICDT, 2003.

[19] P. Paraskevopoulos, *Modern Control Engineering*, Marcel Dekker, 2002.

[20] C. Petkov, *Computational Methods for Linear Control Systems*, Prentice Hall, 1991.