

Provider-Level Content Migration Strategies in P2P-Based Media Distribution Networks

Haiqin Liu, Yan Sun, and Min Sik Kim
 School of Electrical Engineering and Computer Science
 Washington State University
 Pullman, Washington 99164-2752, U.S.A.
 Email: {hliu,ysun,msk}@eecs.wsu.edu

Abstract—In a P2P-based media distribution network (PMDN), content migration is one of the key problems that affect the overall performance of a system. In the system hierarchy, the content migration problem consists of two levels of migration: provider-level migration and user-level migration. In this paper, we focus on the former, provider-level migration, where the goal is to reduce data migration cost while enhancing the local cache hit ratio of media contents. Since this is an NP-complete problem, we propose two types of heuristic migration strategies: object-benefit-based migration (OBM) and peer-benefit-based migration (PBM). The former maximizes the local benefits in allocating a data object to a certain node, while the latter maximizes the benefit of each peer. Experimental results show the effectiveness of both algorithms, which significantly outperform random and round-robin migration schemes.

I. INTRODUCTION

P2P-based media distribution network (PMDN) is built based on the architecture of the traditional content distribution network (CDN). In the distribution network, media data is separately stored and can be shared among the nodes. By such mechanism, a better data distribution service can be achieved in the distribution network.

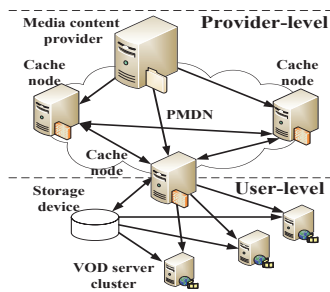


Fig. 1. Architecture of P2P-based media distribution network

Figure 1 shows the typical architecture of such system, which is a two-level scheme. According to the two-level architecture shown in the figure 1, there are two different performance requirements for the content migration problem in each level. For the user-level called streaming media level, which is directly faced by the end-users, a reasonable distribution of the media data across the network will have great impact on the load balance of the related local VOD cluster. For the user-level system, we notate the corresponding migration problem as *user-level content migration*. A series of

research works [1], [2], [3] focused on the *user-level content migration* problem. The basic idea of those studies is to design a cooperative caching mechanism which dynamically adjusts the media content across the servers to maximally satisfy the demands from the end-users.

In this paper, we chiefly focus on the migration problem in the provider-level, which pushes the original content from the media provider server to those cache nodes. Improving the load balance performance among the cache nodes in provider-level is not so urgent. In this level, in order to reduce the frequency of expensive data-fetching from remote cache nodes, the periodical migration of the content among the cache nodes is necessary. As a result, how to reduce the overall cost of the transmission of media content under the bandwidth and the storage constraints and improve the local hit ratio of the requests from those VOD clusters is more attractive. Compared with the *user-level content migration* problem, we call the dispatch and migration problem in this level as *provider-level content migration*. In the following description, if not specifically pointed out, we only refer to the *provider-level content migration* problem.

This paper is structured as follows. Section II presents the related works. In section III, we describe the provider-level content migration problem and establish the mathematical model for this problem. In section IV, based on the model analysis, we propose two heuristic migration algorithms to address this problem. In section V, we conduct several experiments to evaluate our approach, and compare it with the random and round-robin strategies. Finally, we conclude this paper in section VI.

II. RELATED WORKS

Current P2P-based systems are mostly designed for file sharing. Thus, the most important performance they focused on is to maximize file availability or reliability of the system [4]. On the other hand, a substantial amount of works [5], [6] has been done on the strategies for solving data replication problem (DRP) in Grid environments. In [5], the complexity of the static data replication is studied and proved to be a NP-hard problem. Loukopoulos et al. [6] studies the Continuous Replica Placement Problem (CRPP), which is the most similar work compared with our work, and proposed a genetic algorithm (GA) to solve this problem. However, the environment

they focused is the general distributed system, which is not suitable for the PMDN system. For example, they did not take the global access status and the history access behavior of one object, which can greatly affect the overall performance of PMDN as pointed out in the section IV, into accounts. Furthermore, their GA-based scheme is hard, if not impossible, to be executed in a distributed manner.

Unlike other data allocation strategies, we propose two light-weight heuristic algorithms, which can be easily to executed in a distributed way, to reduce the communication overhead of the PMDN system.

III. PROVIDER-LEVEL MIGRATION PROBLEM

In the PMDN, the main goal of the provider-level content migration problem is to migrate media objects among the cache nodes to minimize the system overhead under the bandwidth and storage constraints while the updated content distribution should better adapt to the new content request pattern from the low level VOD cluster.

Consider a distributed peer-to-peer network $S = \{S_1, S_2, \dots, S_m\}$ which contains m peer servers, each node has its own storage capacity and data objects. Let s_i denote the storage capacity of i th node, where $1 \leq i \leq m$. We use $C(i, j)$ to denote the mutual communication cost between node S_i and S_j . Compared with a practical network environment, such cost can be considered as the bandwidth overhead, the transfer delay, or the number of hops between two nodes. We suppose the network is a graph with symmetric communication costs, which means $C(i, j) = C(j, i)$. There are totally n data objects $O = \{O_1, O_2, \dots, O_n\}$ in the network and let o_k denotes the size of the object O_k , where $1 \leq k \leq n$. Let r_k^i and u_k^i denote the object request and update frequency for the object O_k in the i th server during a certain period of time. For each data object O_k , each node S_i maintains an entry which records the IP address information of the nearest node which contains the object O_k . Let N_k^i denote such nearest node for the node S_i . The directory server is responsible for maintaining the global state of the data distribution. Whenever the storage state of one node is changed, such as adding or deleting of one object in this node, the updated storage state will be reported to the directory server. For each node in the network, the directory server maintains a list, which records the information of different N_k^i for different objects. In certain special case, we will have $N_k^i = S_i$, which means the node has the requested object in its local storage. Furthermore, we use AS_k to represent the collection of all the nodes which contain the data object O_k in the local storage. Thus, we have $C(N_k^i, i) = \min\{C(S_k^i, i) | S_k^i \in AS_k\}$, where S_k^i are nodes that contain object O_k .

Consider the small impact on the overall data migration cost of control messages, we do not take their overhead into accounts. Therefore, the overall migration cost contains the migration and the request-update operation of objects in all nodes. The object request cost of the object O_k in the i th server can be defined as:

$$R_k^i = r_k^i \times o_k \times C(i, N_k^i) \quad (1)$$

where $N_k^i = \{S_j | S_j \in S_P^k \cap \min C(i, j)\}$.

The cost for updating object O_k in i th node can be denoted as:

$$U_k^i = u_k^i \times o_k \times \sum_{\forall j \in S_P^k} C(S_O^k, j) \quad (2)$$

where S_O^k is the original content provider node of the object O_k . Therefore, the total flow cost of the whole network according to the object deployment P can be expressed as:

$$D_P = \sum_{k=1}^n \sum_{i=1}^m (R_k^i + U_k^i) \quad (3)$$

Moreover, the cost of object migration in the network can be denoted as:

$$M_{PP'} = \sum_{k=1}^n \sum_{i=1}^m p'_{ik} (1 - p_{ik}) o_k C(i, S_R^k) \quad (4)$$

where S_R^k is the node that contains the data object O_k that we can select according to a object migration plan R . The selection policy R can be random selection strategy or round-robin selection strategy which picks one of neighbor nodes in turn. Both of these two strategies has its own limitations, which will be demonstrated in the experimental results.

Therefore, the overall benefits of the migration can be denoted as:

$$B_{PP'} = D_P - (D_{P'} + M_{PP'}) \quad (5)$$

Given a network topology and the read-update mode, D_P is a constant value.

As a result, the optimization problem can be described as: Given the peer network topology, the original deployment P and the corresponding read-update model for objects, find out an optimal adjustment strategy that meet the related constraints. In other words, the goal is to maximize the value $B_{PP'}$ under certain constraint.

This optimal problem has the following constraints:

- 1) The storage capacity of peer nodes is limited.

$$\sum_{k=1}^n p_{ik} o_k \leq s_i, \quad 1 \leq i \leq m; \quad (6)$$

which means that the sum of the object size stored in node S_i should not exceed the related storage capacity.

- 2) In order to ensure the data availability in the system, there must be at least one replica of each data object stored in the content provider server.

$$p_{S_O^k} = 1, \quad 1 \leq k \leq n; \quad (7)$$

- 3) At most one replica is allowed to be stored in one peer node.

$$p_{ik} \in \{0, 1\}, \quad 1 \leq k \leq n, 1 \leq i \leq m, \quad (8)$$

IV. CONTENT MIGRATION ALGORITHM

In this section, we will propose two heuristic algorithms in order to achieve the approximate optimal global solution for the problem described above.

A. Analysis

Firstly, we divide the overall running time of the system T into equal time periods $T_1, T_2, \dots, T_j, \dots, T_N$. Suppose $P_k = \{p_{k1}, p_{k2}, \dots, p_{km}\}$, a one-dimensional boolean vector, is an object allocation deployment for a particular data object O_k . When the object O_k is replicated in the node S_i , the request cost of this object from the end users will be reduced according to a node selection mechanism. Therefore, the request benefit we can obtain by adjusting the object O_k in node S_i is:

$$\Delta R_k^i = o_k (r_k^i(T_{j+1})C(i, N_k^i(T_{j+1})) - r_k^i(T_j)C(i, N_k^i(T_j))) \quad (9)$$

Where $r_k^i(T_j)$ and $r_k^i(T_{j+1})$ are the request frequency before and after the migration deployment. Similarly, $N_k^i(T_j)$ and $N_k^i(T_{j+1})$ reflect the change of the state of the object O_k in node S_i after the period T_j .

Also, the migration cost for a specific object O_k during the adjustment can be denoted as:

$$T_{P(T_i)P(T_{i+1})}^{ik} = p_{ik}(T_{j+1})(1 - p_{ik}(T_j))o_k C(i, S_R^k) \quad (10)$$

Similarly, the benefit achieved by updating the object O_k can be denoted as:

$$\Delta U_k^i = o_k \left(u_k^i(T_{j+1}) \sum_{\forall l \in S_p^k(T_{j+1})} C(S_O^k, l) - u_k^i(T_j) \sum_{\forall l \in S_p^k(T_j)} C(S_O^k, l) \right) \quad (11)$$

ΔR_k^i and ΔU_k^i are not necessarily positive, while the negative value of any ΔR_k^i or ΔU_k^i indicates that the placement of the object O_k in the node S_i is not reasonable from the local standpoint. However, this does not mean the overall benefits cannot be achieved by sacrificing the local benefits.

Based on the above discussion, we can define the overall benefits, as shown in equation 12, when we place the data object O_k in the node S_i after one migration.

$$OBenefit_{P(T_j)P(T_{j+1})}^{ik} = \Delta R_k^i + \Delta U_k^i - T_{P(T_j)P(T_{j+1})}^{ik} \quad (12)$$

$OBenefit_{P(T_j)P(T_{j+1})}^{ik}$ can be considered as the local benefit obtained by allocating object O_k in the node i during the T_{j+1} period. However, such benefit does not take both the history request frequency information of this node and the global object request information frequency into accounts. Ignoring the former factor will cause objects to be frequently scheduled among different nodes in a short period of time due to the random request pattern of the end users, which will increase the overhead of the data transfer. Also, ignoring the latter factor will result in the consequence that those data objects with extremely low access frequency in the local area, which might have a high popularity in other nodes, will never be scheduled. If so, it will lead those requests for these objects to be frequently forwarded to the content provider server, which will make a bandwidth bottleneck for accessing the original

content and result in a low data availability of the overall system.

Let $f_k^i(T_j) = u_k^i(T_j) + r_k^i(T_j)$ denote the total access frequency of the object O_k . We can redefine the local benefit by introducing $F^{ik}(T_j)$, which contains the information of both global and history access frequency of this object, as define in the equation 13.

$$G_{P(T_j)P(T_{j+1})}^{ik} = OBenefit_{P(T_j)P(T_{j+1})}^{ik} + \alpha F^{ik}(T_j) \quad (13)$$

where

$$F^{ik}(T_j) = \frac{f_k^i(T_j) - f_k^i(T_{j-1})}{T_j} + \beta \sum_{i=1}^m f_k^i(T_j) \quad (14)$$

where α and β are benefit correction and frequency correction factor, respectively. The first item of the right side of the equation 14 reflects the history request trend, and the great value of this value indicates the increasing trend of the access frequency of the object O_k . The second item indicates the global access status of this object. Similarly, the great value of this item means the large number of overall requests for this object in the system during a certain period of time.

B. Algorithm Description

In this section, we propose two types of heuristic migration strategies which are called object-benefit-based migration algorithm (OBM) and peer-benefit-based migration algorithm (PBM), respectively. The OBM method is proposed from the angle of data object while the PBM is considered from the perspective of peer node itself.

1) *Object-benefit-based Migration Algorithm*: We define the unit-object benefit as $U_{P(T_j)P(T_{j+1})}^{ik}$, which represents the benefit that can be obtained when we allocate an unit object O_k in the node S_i . This definition also means the ‘‘value for money’’ level of this unit object.

$$U_{P(T_i)P(T_{i+1})}^{ik} = \frac{G_{P(T_i)P(T_{i+1})}^{ik}}{o_k} \quad (15)$$

Based on this definition, we firstly propose the object-benefit-based migration algorithm for the object migration.

Initially, suppose there is no object in the system except the content provider. According to the statistical information of requests of data objects during a certain time period T , the unit-object benefit $U_{P(T_j)P(T_{j+1})}^{ik}$ can be calculated based on the equation 13 and 15. Then, we allocate the object with the maximal $U_{P(T_j)P(T_{j+1})}^{ik}$ to the node S_i under the storage capacity limit. If the node S_i already contains the object O_k , the corresponding allocation request can be simply discarded by the node. In order to better describe the algorithm, we maintain a linear linked list $ObjectList(i)$ in each node S_i . Each element in the list represents those objects that are allowed to be allocated in the local node. In other words, this list maintains those absent objects which are not in the local storage. Furthermore, we use another linear list $ListServer$ to maintain those nodes that still are able to accept allocation requests of new objects. For each node S_i in the list $ListServer$, we have $ObjectList(i) \neq \emptyset$. Also, we use

$restStore(i)$ to denote the remaining storage capacity of the node S_i . Algorithm 1 shows the pseudo-code description of the OBM.

Algorithm 1: Object-benefit-based Migration Algorithm

```

1 Initialize  $ListServer = S$ ,  $restStore(i) = s_i$ ;
2 for  $i = 1$  to  $m$  do
3    $ObjectList(i) = O$ ;
4 end
5 while  $ListServer \neq \emptyset$  do
6   Select a node  $S_i$  in a Round-Robin method;
7   while  $ObjectList(i) \neq \emptyset$  do
8     for  $k = 1$  to  $\|ObjectList(i)\|$  do
9       Compute each  $U_{P(T_j)P(T_{j+1})}^{ik}$ ;
10       $U_{max}^i \leftarrow \max(U_{P(T_j)P(T_{j+1})}^{ik})$ ;
11    end
12    Select the  $O_k$  with the maximal value  $U_{max}^i$ ;
13    Mark the maximal value object with  $O_{max}^i$ ;
14    if  $U_{max}^i \geq 0$  and  $restStore(i) \geq 0$  then
15      Allocate object  $O_{max}^i$  in the node  $i$ ;
16       $restStore(i) = restStore(i) - o_{max}^i$ ;
17    end
18     $ObjectList(i) = ObjectList(i) - O_{max}^i$ ;
19    for  $i = 1$  to  $m$  do
20      Update all the value of  $N_k^i$ ;
21    end
22  end
23   $ListServer = ListServer - S_i$ ;
24 end

```

2) *Peer-benefit-based Migration Algorithm*: OBM is proposed from the angle of benefits of data objects. Similarly, from the standpoint of benefits of each node, another heuristic migration algorithm PBM can be proposed. Suppose the object distribution deployment in node S_i is $P_i = \{p_{i1}, p_{i2}, \dots, p_{ik}, \dots, p_{in}\}$, we define the node benefit as:

$$peerBenefit(i) = \sum_{k=1}^n p_{ik} \cdot G_{P(T_j)P(T_{j+1})}^{ik} \quad (16)$$

Given a specific node i , there will be many object allocation deployments. Among all these object allocation plans for node i , we can get a object deployment with the maximal peer benefit value according to the equation 16. Thus, for each node, we pre-compute the plan with maximal peer benefit and we notate this process as a sub-problem which will be further analyzed below. Among all these plans with maximal value, we choose the plan with the maximum value and then start to execute the real object allocation process. Then, under the changed status of the global object deployment, we repeat this step again until all the nodes are allocated.

By comparing with the classical knapsack problem, we find that the sub-problem pointed out above is essentially equivalent to the knapsack problem, which is a NP problem. In order to simplify the overall problem, we adopt the greedy algorithm to solve this sub-problem, and approximately consider the resulted object deployment P_i as the optimal solution for this sub-problem. We denote the corresponding maximal node benefit as $peerBenefit(i)^*$.

Algorithm 2 shows the pseudo-code description of the PBM.

Algorithm 2: Peer-benefit-based Migration Algorithm

```

1 Initialize  $ListServer = S$ ,  $restStore(i) = s_i$ ;
2 for  $i = 1$  to  $m$  do
3    $ObjectList(i) = O$ ;
4 end
5 while  $ListServer \neq \emptyset$  do
6   for  $i = 1$  to  $\|ListServer\|$  do
7     for  $k = 1$  to  $n$  do
8       Compute each  $U_{P(T_j)P(T_{j+1})}^{ik}$ ;
9       Select the  $O_k$  with the maximal value  $U_{max}^i$ ;
10      if  $U_{max}^i \geq 0$  and  $restStore(i) \geq 0$  then
11        Preassign the object  $O_{max}^i$  in the node  $S_i$ ;
12         $ObjectList(i) = ObjectList(i) - O_{max}^i$ ;
13         $restStore(i) = restStore(i) - o_{max}^i$ ;
14      end
15    end
16    Compute  $peerBenefit(i)$ ;
17  end
18  Select node  $S_i$  with the maximal value  $peerBenefit(i)^*$ ;
19  Mark that peer with the symbol  $S_{max}$ ;
20  Allocate  $S_{max}$  with the objects by the preassigned deployment  $P_{max}$ ;
21   $ListServer = ListServer - S_{max}$ ;
22  for  $i = 1$  to  $m$  do
23    Update all the value of  $N_k^i$ ;
24  end
25 end

```

3) *Analysis of Algorithms*: Due to the space limit, we directly give the overall time complexity of the above two algorithm here. In the worst case, the time complexity of the proposed two algorithms are $O(n^2m^2)$ and $O(m^2n \log n)$, respectively.

V. EVALUATION

A. Experimental setup

We use MediSyn [7] to generate the update and request log. The generated requests are evenly distributed to m node in the system. Moreover, we simulate all the following experiments by using a P2P simulation tool GPS [8]. We set m peer nodes in the system and each node has the same storage capacity s . The total number of objects in the system is n . The link cost between a pair of nodes follows the uniform distribution. Since we divide the media content into pieces in the system, the size of each object o_k is a constant o . The access pattern of objects follows Zipf distribution, which has Zipf parameter θ_r and θ_w for the request pattern and update pattern. That means the access frequency of an object with the popularity order k is proportional to the $k^{-\theta}$. For a medium-sized network, the total number of requests and updates can be set as 100,000. Table I shows the default setting of parameters adopted in our experiment.

B. Performance Analysis

We compare the proposed two algorithms against both random and round-robin strategy to evaluate the performance.

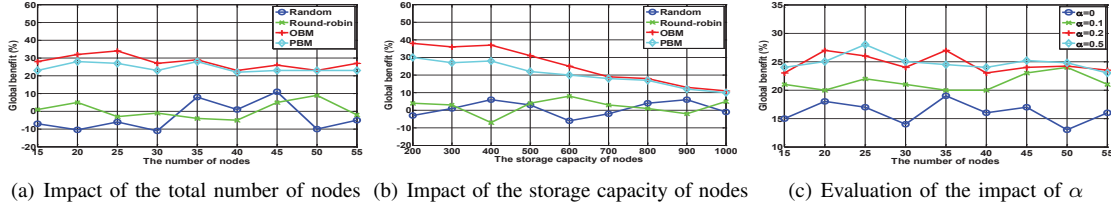


Fig. 2. Detection under various attack rates

TABLE I
THE DEFAULT PARAMETER SETTINGS OF THE EXPERIMENTS

Item	Parameter	Description	Setting value
Node	m	Number of nodes	35
	s	Storage capacity	600
Object	n	Number of objects	500
	o	Object size	1
Link cost	c_{\min}	Minimal value of cost	1
	c_{\max}	Maximal value of cost	10
Zipf	θ_r	Parameter of requests	1.5
	θ_w	Parameter of updates	0.4
Correction factor	α	Benefit correction	0.2
	β	Frequency correction	0.1
Simulation period	Days(day)	Simulated time period	30

Specifically, the random strategy means we randomly choose a candidate node for the data object allocation while the round-robin strategy will choose a candidate according to the corresponding order. We define a key metric called the global benefit for comparison. This metric reflects the saved system overhead after one migration, which is defined as follows.

$$GlobalBenefit_{PP'} = \frac{D_P - (D_{P'} + M_{PP'})}{D_P} \times 100\% \quad (17)$$

Figure 2(a) illustrates the simulation results of different strategies when we vary the number of nodes in the system. From the results, we can draw the following conclusions. Firstly, regarding the saved overhead of the system, the proposed two algorithms greatly outperform the random and round-robin strategy, which both ignore any change of network conditions. Furthermore, the performance difference between the OBM and PBM are only slightly different. When the number of nodes in the system is relatively small, OBM will be slightly superior in performance. That is because OBM is implemented from the perspective of each single object for the content adjustment, which can result in a finer scheduling deployment, while PBM is from the angle of the node benefit, which is a kind of coarser plans. When the size of the system grows, the performance gap between these two algorithms is very small. The reason is that the probability of obtaining needed objects from the neighbor nodes will increase when the objects have been distributed into more nodes in the system. In this case, PBM strategy is more attractive, since its time complexity is much smaller than OBM.

We also evaluate the impact of the node's capacity to the global benefit. Figure 2(b) shows the simulation results. From the results, similarly, we can see that both OBM and PBM are superior to the random and round-robin strategy. Second, OBM performs better than PBM due to the similarly reason

pointed out in the first experimental case. Furthermore, as the storage capacity of nodes grows, the performance gap will decrease. That is because more objects will be allocated into the local storage device, which will also increase the hit ratio of objects in the local area.

Figure 2(c) shows the relationship between parameter α and the global benefit under the PBM scheme. We can see that the introduction of correction parameter is necessary for improving the overall system performance. However, it is not suitable to assign a large value to $alpha$, since a larger value of $alpha$ will not help to obtain more global benefits.

VI. CONCLUSION

In this paper, aimed at the object migration problem in the PMDN system, we firstly establish the model of the migration problem, and then we proposed two heuristic migration algorithms which are called unit-object benefit migration (OBM) algorithm and peer-benefit migration (PBM) algorithm, respectively. Experimental results show the effectiveness of these two algorithms, which both greatly outperform the random and round-robin migration scheme.

REFERENCES

- [1] J. Z. Wang and V. Bhulawala, "Design and implementation of a P2P cooperative proxy cache system," in *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, 2005, pp. 508–514.
- [2] J. Ni and D. H. K. Tsang, "Large-scale cooperative caching and application-level multicast in multimedia content delivery networks," *IEEE Communications Magazine*, vol. 43, no. 5, pp. 98–105, May 2005.
- [3] L. Guo, S. Chen, S. Ren, and X. Chen, "PROP: A scalable and reliable p2p assisted proxy streaming system," in *Proceedings of IEEE International Conference on Distributed Computing Systems*, Tokyo, Japan, Mar. 2004.
- [4] C. Y. Chan, J. Y. B. Lee, and M. Hamdi, "Gradient-descent-scheduler network-aware transmission scheduler for server-less video streaming systems," in *Proceedings of IEEE International Conference on Communications*, May 2005.
- [5] U. Cibej, B. Slivnik, and B. Robic, "The complexity of static data replication in data grids," *Journal of Parallel Computing*, vol. 31, no. 8+9, pp. 900–912, Nov. 2005.
- [6] T. Loukopoulos and I. Ahmad, "Static and adaptive distributed data replication using genetic algorithms," *Journal of Parallel and Distributed Computing*, vol. 64, no. 11, pp. 1270–1285, Nov. 2004.
- [7] T. Wenting, F. Yun, and C. Ludmila, "Long-term streaming media server workload analysis and modeling," HP Laboratories, Technical Report HPL-2003-23, Feb. 2003.
- [8] W. Yang and N. A. Ghazaleh, "GPS: A general peer-to-peer simulator and its use for modeling bittorrent," in *Proceedings of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Atlanta, GA, USA, Sep. 2005.