SW-Uinta: A Small-World P2P Overlay Network*

Jie Xu, Hai Jin

Services Computing Technology and System Lab.
Cluster and Grid Computing Lab.
School of Computer Science and Technology
Huazhong University of Science and Technology, Wuhan, 430074, China
hjin@hust.edu.cn

Abstract. In this paper, we propose a new structured P2P overlay network, named SW-Uinta, where employs a non-deterministic caching strategy that allows for polylogarithmic search time while having only a constant cache size. Compared with deterministic caching strategies proposed by previous P2P systems, the non-deterministic caching strategy can reduce communication overhead for maintaining the routing cache table. Cache entries in the peer can be updated by subsequent queries rather than only by running stabilization periodically. A novel cache replacement scheme is used to improve lookup performance. We compare the performance of our system with that of other structured P2P networks such as Chord and Uinta. It shows that the SW-Uinta protocol can achieve improved object lookup performance and reduce maintenance cost compared with some other protocols.

1. Introduction

P2P systems are self-organizing distributed systems with no centralized control. Each peer in the P2P network has similar functionalities and plays the roles of a server and a client at the same time. These systems have recently gained much attention, primarily because of the great number of features they can offer to applications that are built on top of them, such as scalability, availability, fault tolerance, decentralized administration, and anonymity.

Current P2P systems can be classified into two types, namely unstructured and structured. Unstructured systems like Gnutella [1], KazaA [2] and Freenet [3] are constructed without any regularization on the connectivity among peers and the routing mechanism. For them, the emphases are on fast file retrieval, with no guarantee that files will always be located. In contrast, structured P2P systems such as Chord [4], CAN [5], Pastry [6] and Tapestry [7] follow a predetermined structure. They guarantee that the file will always be located at the cost of increased overhead for peers join/leave and maintaining the routing table. Therefore, the research issue of this paper is whether there exists a scheme such that each file can be located and maintenance cost can be reduced.

* This paper is supported by National Science Foundation of China under grant 60433040, and CNGI projects under grant CNGI-04-12-2A and CNGI-04-12-1D

Most of the existing structured P2P systems adopt the deterministic caching scheme [8], where keys should be addressed in the cache of peer *N* is based on the key of peer *N* and cached index entries typically have expiration times after which they are considered stale. However, little attention has been given on how to maintain these caches during the lookup process. Therefore, communication overhead is high for maintaining the routing cache table.

In this paper, we propose a non-deterministic caching scheme to maintain routing cache tables in a structured P2P overlay network, named Uinta [9]. Our scheme is built on the structured system which can guarantee that each file can be located. The non-deterministic caching scheme builds of cache index entries after answering search queries which can reduce maintenance cost. The basic idea of our scheme is to arrange all peers along a ring-over-ring and equip them with some short distance contacts and long distance contacts. Short distance contacts are built when the peer joins the system with its immediate neighbors. Long distance contacts are built after the peer receives the reply that it requests for.

Assume that peer *S* initiates the query for key *K* which is in the cache of peer *T*. Upon receiving the answer from peer *T*, peer *S* caches the information of peer *T* which arrives with the reply. The traditional cache replacement scheme such as LRU, LFU, and FIFO can not result in the better lookup performance because they do not consider the network topology. In order to optimize the performance of global system, we use the intuition from the small world model [10-14] which says that the routing distance in a graph will be small if each peer has pointers pointing to its immediately neighbors and some chosen far away nodes.

Compared with our previous work [9], main contributions in this paper are:

(1) We propose a non-deterministic caching scheme to reduce maintenance cost for updating the routing cache table.

(2) We propose the SW cache replacement scheme with the small-world paradigm to further improve the performance of object lookup.

The rest of this paper is organized as follows. In Section 2, we give a brief background description of the small-world model and Uinta overlay network. In Section 3, we provide the method how to construct SW-Uinta overlay network. SW-Uinta routing algorithm is proposed and its complexity is given in Section 4. Experiments are discussed and results show the performance of SW-Uinta outperforms that of some other systems in Section 5. An overview of related works is presented in Section 6 and we conclude our research and propose future work in the last section.

2. Background

2.1 Small-World Model

The notion of small world phenomenon originates from social science research by Stanley Milgram [10]. He sought to determine whether most pairs of people in society were linked by short chains of acquaintances. Through some experiments, he

concluded his research by showing that most pairs of people are joined by a median number of six steps, a so-called "six degrees of separation" principle.

A theoretical model for small-world networks by Watts and Strogatz [11] pictured a small world as a loosely connected set of highly connected sub-graphs. The edges of the network are divided into "local" and "long-range" contacts, which are constructed roughly as follows. One starts with a set V of n points spaced uniformly on a circle, and joins each point by an edge to each of its k nearest neighbors, for a small constant k. These are the "local contacts" in the network. One then introduces a small number of edges in which the endpoints are chosen uniformly at random from V — the "long-range contacts". However, according to the model of Watts and Strogatz, there is no decentralized algorithm capable of constructing paths of small expected length [13].

Kleinberg [13] defined an infinite family of network models that naturally generalized the model in [11] and then proved that there was exactly one model within this family for which a decentralized algorithm existed to find short paths with high probability. In this model, the probability of a random shortcut being a distance x away from the source is proportional to $1/x$ in one dimension.

Now, it has been observed that the small world phenomenon is pervasive in a wide range of settings such as social communities, biological environments, and data/communication networks. For example, recent studies [15] have shown that P2P networks such as Freenet may exhibit small world properties. Generally, small world networks can be characterized by average path length between two nodes in the network and cluster coefficient defined as the probability that two neighbors of a node are neighbors themselves. A network is said to be small world if it has small average path length (i.e., similar to the average path length in random networks) and large cluster coefficient (i.e., much greater than that of random networks). Studies on a spectrum of networks with small world characteristics show that searches can be efficiently conducted when the network exhibits the following properties: 1) each node in the network knows its local neighbors, called short range contacts; 2) each node knows a small number of chosen distant nodes, called long range contacts, with probability proportional to $1/x$ where x is the distance.

2.2 Uinta Overlay Network

P2P overlay networks, such as CAN, Chord, Pastry and Tapestry, lead to high latency and low efficiency because they are independent of underlying physical networks. A well-routed lookup path in an overlay network with a small number of logical hops can result in a long delay and excessive traffic due to undesirably long distances in some physical links. In these DHT-based P2P systems, each data item is associated with a key and the key/value pair is stored in the peer to which the key maps, not considering the data semantic. In [9], we propose an effective P2P routing algorithm, called Uinta, to adaptively construct a structured P2P overlay network. Uinta not only takes advantages of physical characteristics of the network, but also places data belonging to the same semantic into a cluster and employs a class cache scheme to consider the users' interest.

Construction of Uinta overlay network involves three major tasks: (1) forming peer clusters based on the physical topology of network; (2) assigning an identifier to a peer or a key to locate a peer in the peer cluster; (3) constructing an overlay network across peer clusters. The detail of Uinta overlay network can be found in [9].

3. SW-Uinta Overlay Network

3.1 Construction

Though experiments have shown that Uinta routing algorithm can improve P2P system lookup performance, a deterministic caching strategy is employed in it, which only achieves *O(logN)* search time with *O(logN)* cache size and maintenance cost for updating the routing table is $O(\log^2 N)$. P2P network is the high dynamic system so that too much maintenance cost will reduce the global performance of system. Now we construct an overlay network SW-Uinta to get $O((\log^2 N)/k)$ search time with *O(k)* cache size. Maintaining the routing table need no additional cost.

In Uinta, each peer maintains two finger tables: *c*-finger table and *l*-finger table, and a class cache table. The deterministic caching strategy is employed for *c*-finger table and *l*-finger table and LRU replacement cache scheme is used for the class cache table. In SW-Uinta, each peer also maintains three cache tables. However, a non-deterministic caching strategy is proposed for two finger tables and a cache replacement scheme related to the small-world model is used for all three cache tables.

In the *c*-finger table of SW-Uinta, each peer maintains two short links: *c-successor* which points to the first-joined peer in the next cluster and *c-predecessor* which points to the first-joined peer in the previous cluster. Each peer maintains *m* long links *c*-finger[*i*] (*1<=i<=m*). In the *l*-finger table of SW-Uinta, each peer maintains two short links: *l-successor* pointing to the next peer in the same cluster and *c-predecessor* pointing to the previous peer in the same cluster. Each peer maintains *m* long links *l*-finger[*i*] (*1<=i<=m*).

If the cache size for peer *P* is *m* and its cache table is full, assuming that the cache table $CT=\{d_1, d_2, \ldots, d_m\}$ and d_m is the distance between the cache object and *P*, the cache object with distance d_i is replaced by the new object with the probability $\frac{1}{D} * \frac{1}{d_{m+1}}$ where $D = \sum_{i=1}^{m+1} \frac{1}{d_i}$ when a new object with distance d_{m+1} is received. We call this scheme SW cache replacement scheme.

Suppose that peer *S* gets the answer requested from peer *T* and there are no pointers to peer *T* in peer *S*.

(1) If peer *S* and peer *T* are in the same cluster and *l*-finger[*i*] is not full, peer *S* caches peer *T* in the *l*-finger table;

(2) If peer *S* and peer *T* are in the same cluster and *l*-finger[*i*] is full, SW cache replacement scheme is employed for *l*-finger[*i*];

(3) If peer *S* and peer *T* are in the different clusters and *c*-finger[*i*] is not full, peer *S* caches peer *T* in the *c*-finger table;

(4) If peer *S* and peer *T* are in the different clusters and *c*-finger[*i*] is full, SW cache replacement scheme is employed for *c*-finger[*i*].

Because users always retrieve data of a kind, which they are interested in, we store the data information based on data semantics in Uinta, which makes data of a kind placed in the same cluster in Uinta. After that, the user can utilize a *class cache table* to cache the identifier of peer where data of some kind searched recently store and the identifier of this kind. If the user searches data of this kind next, it can use the information of the cache table directly. It is obvious that P2P system workload has temporal and spatial localities just as that in the web traffic [16]. For example, a user who retrieves a song is likely to retrieve other songs in subsequential requests. A high class cache table hit rate can be expected, thus a reduced average number of routing hops and lower routing network latency can be achieved.

In Uinta, we used LRU as the cache replacement scheme, which could not improve the whole performance because the topology of network does not be considered in LRU. Therefore, in SW-Uinta, we also employ the SW cache replacement scheme. When peer *S* gets data *D* requested from peer *T*, the class identifier of data *D* and the IP address (and port number) of the first-joined peer in the cluster where *T* is will be stored in the class cache table if it is not full. Otherwise, SW cache replacement scheme is employed for the class cache table.

Cache tables are generally kept fresh by the traffic of requests traveling through peers. To handle pathological cases in which there are no lookups for a particular ID range, each peer refreshes any cache table to which it has not performed data lookup in the past hour. Refreshing means picking an entry in the cache table and performing a peer search for that ID. On the one hand, this scheme avoids the bottleneck of network traffic because all of the peers will not update at the same time. On the other hand, the peer can be responsible for the failure of some peers quickly. Maintenance cost for the routing table always goes with the lookup operation, which needs few other messages.

3.2 Peer Operation

Peer joins. When a new peer *p* joins the system, it sends a join message to a nearby peer *q* that is already a member of the system. This process can be done in different methods. We simply assume it can be done quickly (this is the same assumption as other DHT algorithms). Then peer *p* can get the information of landmark nodes from this nearby peer *q* and fulfill its own landmark table. It then decides the distance between itself and the landmark nodes and uses the distributed binning scheme to determine the suitable cluster P_p it should join. Then the identifier D_p of peer *p* can be gotten, *i.e.,* $D_p=P_p*2^n+S_p$ (S_p is the hash value of IP address of peer *p*). Consequently, peer *p* connects the peer *p'* in the cluster P_p through the *c*-finger table of peer *q* and then is located in the cluster based on the suffix S_p. Assume that peer *s* is the *l-successor* of peer *p* and peer *n* is the original *l-predecessor* of peer *s*. Then, peer *p* acquires peer *s* as its *l-successor* and acquires peer *n* as its *l-predecessor*. Peer *n* acquires peer *p* as its *l-successor* and peer *s* acquires peer *p* as its *l-predecessor*. Other data structures needed by peer *p* are copied from peer *s*.

If peer p finds that *c-finger[i].identifier* equals to P_p but the identifier prefix of *c-finger[i].node* denoted as peer x does not equal to P_p rather than X_p in peer q, it shows peer p will form a new cluster whose identifier is P_p. Peer p acquires peer x as its *c-successor* and acquires peer q as its *c-predecessor* which is the original *c-predecessor* of peer x. Every peer in the cluster that peer x located, when notified by peer p, acquires peer p as its *c-predecessor*. Every peer in the cluster that peer q located acquires peer p as its *c-successor*. Both *l-predecessor* and *l-successor* of peer p point to itself. Other data structures needed by peer p are copied from peer s. Finally, keys between P_p*2^n and X_p*2^n are moved form cluster X_p*2^n to cluster P_p*2^n. Peer p joins the system successfully.

Peer leaves or fails. When a peer leaves the network, it checks whether it is the last peer in the cluster. If there are other peers in the cluster, this peer simply informs its leaving to its *l-predecessor* and *l-successor* and keys in it are moved to its *l-successor*. Otherwise, except for informing its leaving to its *c-predecessor* and *c-successor*, the key subspace of this cluster needs to be merged with one of its neighboring clusters. The peer of its neighboring clusters which is closest in the key space to the leaving peer takes over all of its keys. A failed peer is detected during routine operations such as search. If a peer detects a failure in one of its cache tables, it evicts this entry in the cache table.

4. SW-Uinta Routing Algorithm

SW-Uinta routing algorithm and its complexity are described in this section.

1) When peer p wants to obtain the file associated with a key k and a class c, it gets the class identifier P_k of the file hashed by SHA-1 with c;
2) Check whether exists an entry (P_k*2^n, q) for the class identifier P_k in the class cache table; if does, jump to peer q directly, then to 6); otherwise, to 3);
3) Check whether P_k falls between the P_p of p and the P_q of its *c-successor* q; if does, jump to q, then to 6); otherwise, to 4);
4) $x=p$;
   repeat
      Search peer x's *c-finger table* for peer q whose prefix of identifier P_q most immediately precedes P_k, $x=q$;
   until P_k falls between the P_x of x and the P_q of its *c-successor* q;
5) Jump to peer q;
6) Find a peer d through the *l-finger table* of peer q so as to make the suffix of key identifier S_k hashed by SHA-1 with k falls between S_x of x and S_d of its *l-successor* d;
7) Return the identifier of peer d and (key, value) pair searched to peer p; join the information of peer d to two finger tables of peer p and join (P_k*2^n, d) to the class cache table of peer p using the SW cache replacement scheme described above.

In SW-Uinta, the expected number of hops required to lookup an object is $O((\log^2 N)/k)$.

5. Performance Evaluation

5.1 Simulation Methodology and Performance Metrics

In our simulation, we use the GT-ITM [17] transit stub topology generator to generate the underlying networks, where the number of system peers N varies from 1,000 to 10,000. As far as the logical overlay is concerned, we build SW-Uinta and Uinta based Chord simulator. Each peer on the overlay is uniquely mapped to one node in the IP layer. We choose 4 landmarks placed at random and there is 3-level latency from landmarks to peers. 100*N pseudo file-ids that are classified 100 kinds are generated and distributed across all the peers in simulated networks. For each experiment, 100,000 randomly generated routing requests (including file-id and its class) are executed. To obtain a fair comparison, we keep the size of the cache table in SW-Uinta to *O(logN)*. We choose Chord as the platform because the ring geometry allows the greatest flexibility.

We consider four metrics to verify the effectiveness of SW-Uinta: (1) Routing hop: the average number of logical hops traversed by search messages to the destination; (2) Routing latency: the average time for search messages from the source to the destination; (3) Latency stretch: the ratio of the average latency on the overlay network to the average latency on the IP network; (4) Maintenance cost: the average number of messages incurred for each peer joins/leaves and for RT maintenance cost which is the average number of messages needed for maintaining the routing table to be up-to-date.

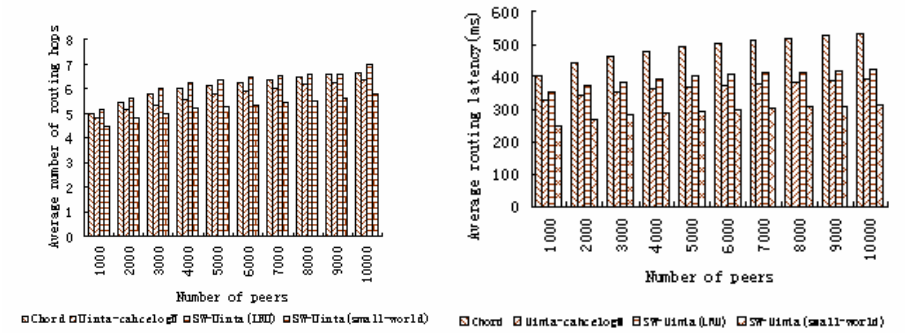
5.2 Routing Cost Reduction

The goal of simulation in this section is to show whether SW-Uinta can reduce the routing cost in P2P system just as Uinta. Fig.1 shows the results of routing hops and routing latency evaluation. In this simulation, we compare the routing performance of Uinta-cache*logN*, SW-Uinta(LRU), and SW-Uinta(small-world) with Chord under different network size.

Fig.1(a) shows the routing performance comparison result measured with the average number of routing hops. All of Chord, Uinta-cache*logN*, SW-Uinta(LRU) and SW-Uinta(small-world) have good scalability: as the network size increases from 1000 nodes to 10000 nodes, the average number of routing hops only increases around 33%, 32%, 36%, and 29%, respectively and the average number of routing hops is 6.07, 5.74, 6.24, and 5.24, respectively.

As a proximate metrics, the average number of routing hops cannot represent the real routing cost. The actual routing latency is highly depended on the average latency for each hop. Fig.1(b) shows the measured results of the average routing latency in Chord, Uinta-cache*logN*, SW-Uinta(LRU), and SW-Uinta(small-world) algorithms. Although Uinta and SW-Uinta have the nearly equal average number of routing hops with that of Chord, they have smaller average routing latency which can represents the real routing cost. For Uinta-cache*logN*, SW-Uinta(LRU), and SW-

Uinta(small-world), the average routing latency gets 24.3%, 18.0%, and 40.4% reduction respectively compared with Chord.



(a) Average number of routing hops (b) Average routing latency

Fig.1. Routing performance comparison for Uinta, SW-Uinta, and Chord

Obviously, the performance of routing hops and scalability for SW-Uinta(LRU) are worse than those of other algorithms, which shows the LRU cache replacement is not suitable for the P2P system. At the same time, because both the network characteristics and the network topology are considered in constructing the SW-Uinta(small-world) system so that the routing performance of SW-Uinta(small-world) is better than that of three other algorithms.

5.3 Stretch Reduction

Latency stretch is referred to the ratio of the average latency on the overlay network to the average latency on the IP network, which can be used to characterize the match degree of the overlay to the physical topology. Table 1 summarizes the stretch statistics in the case of a 10,000 peer network. From the table, we can find that stretch can be reduced using Uinta and SW-Uinta. This shows that using topology-aware and semantic-aware overlay construction, we can achieve significant improvement in the lookup performance. SW-Uinta(small-world) can get better performance than Uinta.

Table 1. Latency stretch result for Chord, Uinta, and SW-Uinta

Algorithm	Average routing latency	Latency stretch
Chord	531.51	4.40
Uinta-cache*logN*	366.68	3.04
SW-Uinta(LRU)	397.28	3.29
SW-Uinta (small-world)	289.31	2.40

5.4 Maintenance Cost

In this experiment, we show that SW-Uinta not only keeps the strengths of Uinta, but also is able to reduce maintenance cost. Maintenance cost here includes two parts: the

number of messages incurred for each peer joins/leaves and RT maintenance cost. Maintenance cost is referred to the average number of messages for all maintaining operations such as peer joining, peer leaving and keeping the routing table to be up-to-date.

We assume that peer joins and leaves according to the Poisson process at rate R=0.1(minute^{-1}). RT maintenance period is an hour. Fig.2 depicts maintenance cost comparison for the Uinta-cache*logN*, SW-Uinta(LRU), SW-Uinta(small-world), and Chord schemes under different network size. Maintenance cost is around 16, 7, 6, and 21, respectively for the Uinta-cache*logN*, SW-Uinta(LRU), SW-Uinta(small-world), and Chord schemes. From this experiment, we know maintenance cost for SW-Uinta is lower than two other algorithms because there is no additional cost for the route table maintenance, that is RT maintenance cost, in SW-Uinta. Each searching operation can update the routing table, which needs no other operations to maintain the routing table to be up-to-date.

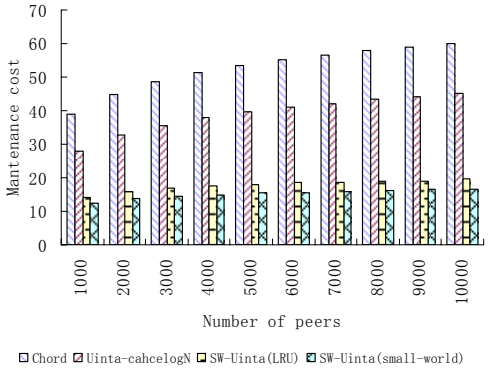


Fig.2. Maintenance cost comparison for Uinta, SW-Uinta, and Chord

6. Conclusions and Future Work

In this paper, we propose a new overlay infrastructure SW-Uinta based on Uinta proposed by us in our previous study. SW-Uinta not only holds the strength of Uinta, which takes both the user's interest and the physical topology into consideration, but also considers the network characteristic so that the SW cache replacement scheme is proposed to further improve the performance of object lookup. Because the P2P system is a dynamic environment, maintenance cost for peer joining, peer leaving and routing state maintenance is very high. Therefore, we propose a non-deterministic caching scheme to reduce maintenance cost when peers join/leave and self-organization occurs. Simulations also show SW-Uinta can improve the lookup performance as well as it can reduce maintenance cost under the same size of routing table. Now, this infrastructure is only suitable for key-based retrieval and content-based retrieval is our next step of work.

Reference

[1] Gnutella: http://www.gnutellaforums.com/

[2] N. Leibowitz, M. Ripeanu, and A. Wierzbicki: Deconstructing the Kazaa Network. *Proc. of 3rd IEEE Workshop on Internet Applications*, Santa Clara, CA, (2003) 112-120

[3] I. Clarke, O. Sandberg, B. Wiley et al: Freenet: A Distributed Anonymous Information Storage and Retrieval System. *Proc. of Workshop on Design Issues in Anonymity and Unobservability*. ICSI, (2000) 311-320

[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *Proc. of the ACM SIGCOMM*. ACM Press, (2001) 149-160

[5] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker: A Scalable Content-Addressable Network. *Proc. of ACM SIGCOMM*. ACM Press, (2001) 161-172

[6] A. Rowstron and P. Druschel: Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems. *Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms*. Springer-Verlag, (2001) 329-350

[7] B. Y. Zhao, L. Huang, J. Stribling, J. Rhea, S. C. Joseph, and A. D. Kubiatowicz: Tapestry: A Resilient Global-Scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications.* Vol.22, (2004) 41-53

[8] N. Sarshar, V. P. Roychowdhury: A Random Structure for Optimum Cache Size Distributed Hash Table (DHT) Peer-to-Peer Design. http://www.ee.ucla.edu/~nima/Publications /opt_cache.pdf, 2002

[9] H. Jin, J. Xu, B. Zou, and H. Zhang: Uinta: A P2P Routing Algorithm Based on the User's Interest and the Network Topology. *Distributed Computing - Lecture Notes in Computer Science*, Springer-verlag, (2005) 238-249

[10] S. Milgram: The Small World Problem. *Psychology Today,* Vol.2, (1967) 60–67

[11] D. Watts and S. Strogatz: Collective Dynamics of Small-World Networks. *Nature* 393, (1998) 440-442

[12] J. Kleinberg: Small-World Phenomena and the Dynamics of Information. *Proceedings of Advances in Neural Information Processing Systems*. MIT Press, (2002) 14-25

[13] J. Kleinberg: The Small-World Phenomenon: An Algorithmic Perspective. *Cornell Computer Science Technical Report* 99-1776, (2000)

[14] A. Iamnitchi, M. Ripeanu, and I. Foster: Small-World File-Sharing Communities. *Proceedings of IEEE INFOCOM,* (2004) 175-186

[15] H. Zhang, A. Goel, and R. Govindan: Using the Small-World Model to Improve Freenet Performance. *Proceedings of IEEE INFOCOM 2002*, (2002) 1228-1237

[16] A. Mahanti: Web Proxy Workload Characterization and Modeling. Master Thesis. Department of Computer Science, University of Saskatchewan, (1999)

[17] E. W. Zegura, K. Calvert, and S. Bhattacharjee: How to Model an Internet Work. *Proceedings of INFOCOM'96*. (1996) 594-602