SAT-Match: A Self-Adaptive Topology Matching Method to Achieve Low Lookup Latency in Structured P2P Overlay Networks *

Shansi Ren, Lei Guo, Song Jiang, and Xiaodong Zhang
Department of Computer Science
College of William and Mary
Williamsburg, VA 23187-8795
{sren, lguo, sjiang, zhang}@cs.wm.edu

Abstract

A peer-to-peer (P2P) system is built upon an overlay network whose topology is independent of the underlying physical network. A well-routed message 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 this paper, we propose an effective method, called SAT-Match, to adaptively construct structured P2P overlay networks, aiming at significantly reducing the lookup routing latency. In this method, each joining peer is initially guided to find a physically close neighbor to connect with. After then, its overlay location is adaptively adjusted whenever a location mismatch is detected. The topology matching optimization in our method solely relies on local neighborhood information. Compared with existing topology matching methods, our method addresses their three limitations: (1) heavily relying on global information about the Internet by using landmark-based measurements, (2) lacking adaptation to frequent peer movement in a dynamic environment, such as mobile networks, and (3) insufficiently accurate in topology matching due to the lack of adaptive topology adjustment. We have evaluated our method in the Content-Addressable Network (CAN), a representative structured P2P system with a strong tolerance to frequent peer arrivals/departures. Through intensive simulation experiments on large scale CAN overlays, we have shown the effectiveness of SAT-Match. Our method can achieve average logical/physical link latency reduction rate by up to 40% . It also outperforms "landmark binning", a method utilizing global information by up to 20%. Finally, combining with the landmark binning method, SAT-Match can achieve up to 60% latency reduction.

*This work is supported in part by the U.S. National Science Foundation under grants CCR-0098055 and ACI-0129883.

1 Introduction

In a Peer-to-Peer (P2P) system, each end node provides services to other participating nodes as well as receives services from them. The central directory based P2P systems such as Napster [7] prevailed in the early stage. Shortly the decentralized unstructured P2P systems such as Gnutella [2] emerged and became popular. However, the tremendous traffic caused by the flooding search mechanism [4] would narrow the scope of its applications. Addressing some limitations of directory-based and Gnutella-like P2P systems, decentralized structured P2P systems based on Distributed Hash Tables (DHTs) make the information search highly objective. CAN [8], Chord [10], Pastry [1] and Tapestry [14] are several representative structured P2P systems. In such systems, pairs of keys and values are placed in an organized way. Keys are hashed first. Based on the hashed values, these key-value pairs are mapped to the corresponding nodes whose IDs match the hashed keys. DHTs contain these distributed, but controlled and organized key-value pairs. Structured P2P systems have the advantages of moderate overhead traffic and easy manageability. Among these representatives, Pastry, Tapestry, and Chord have $O(\log n)$ routing hops, while CAN has $O(dn^{\frac{1}{d}})$ routing hops, where n is the number of peer nodes, and d is the dimension degree of the overlay space. The number of hops for CAN can be nontrivial when d is small.

P2P systems are constructed in overlay networks at the application layer without taking the physical network topologies into consideration. The mismatch between physical topologies and logical overlays is a major factor that delays the lookup response time, which is determined by the product of the routing hops and the logical link latency. Although structured P2P systems can have a relatively moderate number of routing hops, the lookup performance can be heavily degraded when several logical hops consist of slow physical links due to long distances or traffic jams.

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

There are two representative studies to address the mismatch problem in structured P2P systems. One is called landmark binning [8], [9], where all nodes measure their distances by communication latencies to several well-known and relatively stable sites called landmarks in Internet. Each node is labeled by the measurement with a sequence value reflecting its average physical distance to the landmarks. Nodes that have the same landmark sequence values are clustered into one single bin. There are $m!$ bins when m nodes are chosen as landmarks in the system. One disadvantage of this landmark scheme is related to the additional burdens to the landmark sites. Because measurement loads on landmarks would increase drastically as the P2P system size increases. Thus, landmark sites can easily become overloaded and are exposed to attacks. Moreover, this scheme is coarse-grained and have difficulty to distinguish relatively close nodes. In the worst case, all nodes could be clustered into one single bin. Another method is to build a global map to help choose shorter routing paths [12], which combines the landmark binning method and small scale RTT measurements to generate the proximity information of nodes. This global proximity state is made available in public and other nodes can look up close neighbors to route to. However, this greedy routing method may not always reduce the lookup response time, especially when there is a significant mismatch between the overlay and physical networks. On the other hand, it does not well adapt to dynamic environments because each node maintains many neighbor states at different layers, which makes the content migration costly. Neither method makes a continuing effort to remap the overlay structure adaptively after a peer is initially accepted and connected in the P2P system.

Reducing the average logical link latency can significantly reduce the lookup response time, which requires an overlay link to closely match its shortest IP route. Desired overlay construction algorithms correct the mismatch by selecting topologically close peers as neighbors. These algorithms need to address several fundamental issues: (1) It should be decentralized and scalable. (2) It should be accurate and adaptive to dynamically changing environments. (3) It should be of low cost.

In order to address the limitations of the above cited work, we propose a method, called SAT-Match that adaptively changes the overlay structure of the P2P system to match the underlying physical topology. Nodes conduct lightweight probing and make selective jumps (location adjustments) accordingly. By iteratively reducing the average logical link latency, the average response time of lookup routing is reduced. Moreover, it can be easily implemented on existing infrastructures like CAN without introducing new operations. Each node maintains only a small number of neighbor states. Through intensive simulation experiments on large scale CAN overlays, we have shown the effectiveness of SAT-Match. Our method can achieve average logical/physical link latency reduction rate by up to 40% . It also outperforms "landmark binning", a method utilizing global information by up to 30%. Finally, combining with the landmark binning method, SAT-Match can achieve up to 60% latency reduction rate.

Compared with the landmark based methods, the proposed SAT-Match has several merits. First, it only utilizes local information, which avoids overloading the landmark sites and the single point of failure problem. Second, the self-adaptation nature of our method not only improves the matching accuracy between the overlay layer and physical networks, it can also well adapt to the dynamically changing physical topologies such as mobile networks. Finally, in SAT-Match, each node can adaptively adjust its probing frequency to reduce the traffic overhead.

This paper is organized as follows. In section 2 we describe SAT-Match in detail. Simulation methodology and performance evaluation are discussed in section 3. We overview and compare some other related work in section 4. Finally we conclude the work in section 5.

2 Self-Adaptive Topology Matching

2.1 The Mismatch Problem and Our Approach

We take a torus overlay structure in CAN system as an example to explain our method, whose physical topology and logical overlay are shown in Figures 1 and 2 respectively.

Starting from the overlay network in Figure 2, we show how the mismatch can be improved through self-adaptation. If node A wants to route a message to node B, it needs to traverse the path A-C-B with the latency of 2+9+9 =20, or to traverse the path A-D-B with the latency of 12+3+9 =24, both of which are measured by latencies of physical links in Figure 1. However, node A could have routed the message to node B at the cost of 2 because there is a direct physical link with latency of 2 between the two nodes. Because of the topology mismatch, a query usually traverses some unnecessary links before reaching its final destination. In Figure 2, after changing the overlay structure on the left to the right one, the logical overlay can perfectly match the underlying physical topology. Compared to the two alternative routing latencies of 20 and 24 for node A to node B, the new routing latency is minimized to 2.

We have designed a method called SAT-Match to adaptively change the overlay network connections following the ideal topology matching scenario in Figures 1 and 2 for the purpose of reducing the average latency of logical hops. As soon as a node joins the system, it probes nearby nodes for distance measurements (the probing phase), and

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

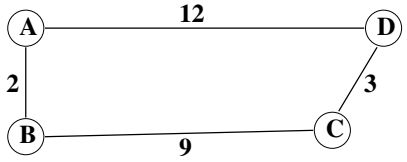


Figure 1. An example of a physical topology with 4 nodes.

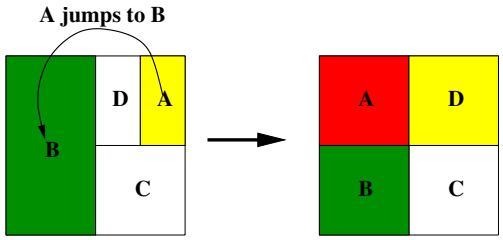


Figure 2. Adjusting overlay connections adaptively to match the logical overlay and its physical network.

picks the closest zone accordingly to jump to (the jumping phase). This iterative process completes until it is close enough to the zone where all its physically close neighbors are located and no additional optimization is necessary. This continuously adaptive topology matching process will achieve a global topology matching optimization in a sufficiently large scope.

One unique advantage of SAT-Match is that no operations other than those in the original CAN protocol are introduced. Since each node in CAN only maintains states of a limited number of neighbors, it makes the *join* and the *leave* operations less expensive than in Chord, Pastry or Tapestry due to a small amount of migration of the states. Furthermore, we can implement the *jump* operation as the combination of a *leave* and a *join* in CAN.

Because the physical topology does not change over times, we define *stretch* as the ratio of the average logical link latency over the average physical link latency to quantify the *topology match degree*, where we refer the *logical link* to the virtual link between a node and one of its direct neighbors in the overlay.

2.2 TTL-k Flooding and RTT Measurements

A recent study shows that flooding with a low number of TTL hops is highly effective, which produces few redundant messages [4]. The probing process of SAT-Match utilizes this effective flooding. Having joined the system based on a DHT assignment, a new node begins probing its neighborhood periodically. The source node floods out a message containing the source IP address, the source timestamp and a small TTL value k to all its neighbors on the overlay. Any node that receives this message responds to the source with its IP address and decrements the TTL field in the messages by 1. If the updated TTL value does not reach 0, this responding node forwards the message to its neighbors.Since k is small, in the end of this flooding, the message covers only a limited number of nodes in a small region. We define these nodes being covered as the *TTL-k neighborhood* of the source node. Specifically, the TTL-1 neighborhood of a node refers to this node and all its direct neighbor nodes in the overlay.

Figure 3 is an illustration of a CAN overlay with 14 nodes, where node 2 is a source that sends TTL-2 probing queries. These messages are flooded and relayed along solid arrows to all nodes that are within 2 hops from the source. The nodes inside the lightly shaded area in Figure 3 are node 2's TTL-2 neighborhood.

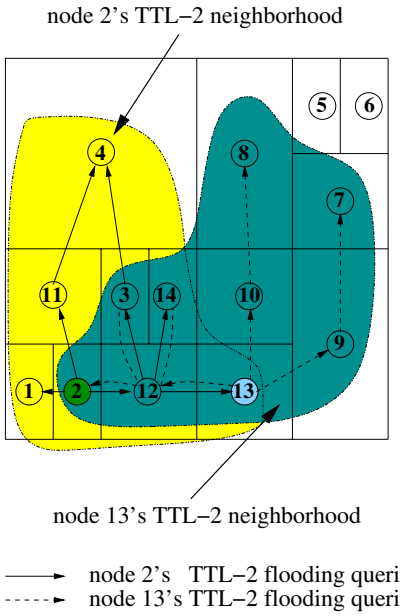


Figure 3. Neighborhoods around node 2 and node 13 on a 14 nodes CAN system.

After probing in a TTL-k neighborhood and collecting a list of IP addresses, the source node uses a ping facility to measure the Round-Trip-Times (RTTs) to each of the nodes that have responded. Then it sorts these RTTs and selects two nodes with the smallest RTTs, based on which, the source node will select one zone associated with one of the nodes to jump in.

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

2.3 Node Jump and Neighborship Adjustment

The ultimate goal of SAT-Match is to group those physically close peer nodes together. For this reason, the source node should choose those peers with the smallest RTTs to it as neighbors. However, simply jumping to a zone where the closest node is found by probing does not always guarantee a stretch reduction for the following reason. Although one node moves towards a physically closer node, it may connect to some new physically distant nodes because of the structural constraints in structured P2P. Thus the stretch reduction obtained from this single direction jump could be offset by latency increases from other new connections. To avoid this problem, we make a precomputation to determine if we make such a jump. A node will not change its current overlay location until some criteria are satisfied to ensure a local optimization of the jump.

Starting from an example, we will then discuss the jumping criteria. In this example, node A has detected its closest node B and its second closest node C in its TTL-k neighborhood. The first attempt of node A is to group itself with node B. Since SAT-Match aims at reducing the stretch of the whole system by each local optimization, node A needs to collect the states of node B and all of its neighbors for a local stretch calculation. By comparing its own stretch with the stretch of node B, node A makes a decision of jumping or not.

For detailed calculation, we use the following definitions and notations.

1) t_0 is the time when node A sits in its current location, and t_1 is the time when node A jumps to the zone associated with node B.

2) $N_{t_0}(A)$ and $N_{t_1}(A)$ are defined as node A's TTL-1 neighborhoods before the jump and after the jump, respectively.

3) $P_{t_0}(A)$ and $P_{t_1}(A)$ are the total numbers of logical neighbor pairs inside $N_{t_0}(A)$ and $N_{t_1}(A)$ respectively.

4) $AL_{t_0}(A)$ and $AL_{t_1}(A)$ are the accumulated latency value of all logical links inside node A's TTL-1 neighborhood before the jump and after the jump, respectively.

5) $L^{avg}_{t_0}(A)$ and $L^{avg}_{t_1}(A)$ are the average logical link latencies in $N_{t_0}(A)$ and $N_{t_1}(A)$, respectively.

For $t \in \{t_0, t_1\}$, we further define the average logical link latency at time t (either t_0 or t_1) for the TTL-1 neighborhoods of both node A and node B.

$$L^{avg}_t(A \cup B) = \frac{AL_t(A \cup B)}{P_t(A \cup B)}$$

If condition of $L^{avg}_{t_0}(A \cup B) > L^{avg}_{t_1}(A \cup B)$ exists, the stretch of A's and B's TTL-1 neighborhoods are reduced.

If $L^{avg}_{t_0}(A \cup B) \leq L^{avg}_{t_1}(A \cup B)$, node A then considers the second closest node C. Similarly, node A calculates $L^{avg}_{t_0}(A \cup C)$ and $L^{avg}_{t_1}(A \cup C)$. If $L^{avg}_{t_0}(A \cup C) > L^{avg}_{t_1}(A \cup C)$, node A jumps to the zone associated with node C. No action is taken in other cases. Our experiments show that targeting the two zones associated with the two closest nodes is most effective. This is because adaptive and iterative local optimizations will quickly or eventually group the physically close nodes in the same overlay zone. If a node outside a zone finds a close node in the zone, the rest of the nodes will be likely to be or eventually to be close to it.

In summary, only when the local average stretch of the source's (node A's) and the sink's (node B's or node C's) TTL-1 neighborhoods is reduced, can a jump take place.

If node X decides to jump to the zone associated with node Y by satisfying the above criteria, The following operations are performed. First, node X leaves its original location and returns its Cartesian zone denoted as $Z(X)$, and (key, value) indices to one of its neighbors. After then, node X contacts node Y to retrieve its Cartesian zone $Z(Y)$. Node X also randomly picks a point P inside $Z(Y)$ as its initial point. Node X sends P's coordinates and a join request to a bootstrap node. The bootstrap node routes the request to zone $Z(Y)$ eventually. Node Y divides its zone $Z(Y)$ into halves and gives one half to node X. Indices owned by Y are migrated at the same time.

2.4 Effectiveness of Node Jump

Here we analyze the system stretch reduction rate for a jump of node X to the zone associated with node Y in the CAN system. Recall that t_0 is the time before node X jumps, t_1 is the time after node X jumps. We define W as the set of all nodes in this system. If

$$L^{avg}_{t_0}(X \cup Y) > L^{avg}_{t_1}(X \cup Y)$$

then with a high probability,

$$L^{avg}_{t_0}(W) > L^{avg}_{t_1}(W).$$

In other words, the reduction of stretch of X's and Y's TTL-1 neighborhoods results in the reduction of stretch of the whole system.

For $t \in \{t_0, t_1\}$, we denote $\overline{N_t(X)} = W - N_t(X)$ thus $\overline{N_t(X \cup Y)} = W - N_t(X \cup Y)$. Because $|W|$ is huge, we can assume that $AL_t(W) = AL_t(\overline{N_t(X \cup Y)})$ and $P_t(W) = P_t(\overline{N_t(X \cup Y)})$.

Recall that the jump criterion for the node X is

$$\frac{AL_{t_0}(X \cup Y)}{P_{t_0}(X \cup Y)} > \frac{AL_{t_1}(X \cup Y)}{P_{t_1}(X \cup Y)}.$$

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

For the experiments we have conducted, with the aid of CAN background zone reassignment algorithm [8], for two typical nodes X and Y, we observed that with 60% probability $P_{t_0}(X \cup Y) = P_{t_1}(X \cup Y)$, and with 20% probability $|P_{t_0}(X \cup Y) - P_{t_1}(X \cup Y)| = 1$. Therefore $P_{t_0}(X \cup Y) \approx P_{t_1}(X \cup Y)$, which implies $AL_{t_0}(X, Y) > AL_{t_1}(X, Y)$. Thus, we have

$$\begin{aligned} & L_{t_0}^{avg}(W) \\ = & \frac{AL_{t_0}(\overline{N_{t_0}(X \cup Y)}) + AL_{t_0}(X \cup Y)}{P_{t_0}(\overline{N_{t_0}(X \cup Y)}) + P_{t_0}(X \cup Y)} \\ = & \frac{AL_{t_1}(\overline{N_{t_1}(X \cup Y)}) + AL_{t_0}(X \cup Y)}{P_{t_0}(\overline{N_{t_0}(X \cup Y)})} \\ > & \frac{AL_{t_1}(\overline{N_{t_1}(X \cup Y)}) + AL_{t_1}(X \cup Y)}{P_{t_1}(\overline{N_{t_1}(X \cup Y)}) + P_{t_1}(X \cup Y)} \\ = & L_{t_1}^{avg}(W). \end{aligned}$$

This agrees with our previous claim. Our experimental results to be presented will further confirm this conclusion.

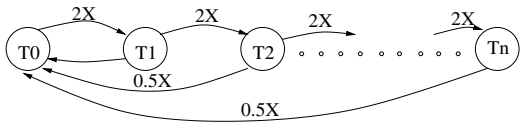
2.5 Nodes Coordination during Self-Adaptation

The self-adaptation for topology match is conducted simultaneously from multiple nodes. When a source node probes and collects statuses of nodes in a targeted zone, all the nodes should not change its status until the probing and jumping process complete. This will ensure the accuracy of topology matching. We provide a coordination mechanism in SAT-Match to synchronize self-adaptations in related zones.

The coordination is based on a query-response model that operates as follows. For node A that targets on a zone associated with node B, after it calculates the local stretch reduction and is ready to jump, it broadcasts a synchronization (SYN) message to all nodes in that TTL-k neighborhood. If those nodes do not receive other SYN messages except this one, they acknowledge node A. Once node A receives acknowledgments from all nodes in its TTL-k neighborhood, it starts to jump to the zone. If A has not received all responses after a period of time, T_{base}, the jumping attempt is aborted.

2.6 Adaptively Varying the Checking Period

After a certain number of iterative probing and jumping steps in each node, the stretch reduction may not be improved anymore. Under such a circumstance a frequent probing causes unnecessary overhead and wastes a lot of bandwidth. Based on a Markov chain model, we deploy an adaptive mechanism to set probing periods for each node at a given time. Once a node has probed its neighborhood and does not jump, its probing period is doubled. Otherwise the period is set to T_0, which is the base probing period. The process is illustrated in Figure 4.



The Markov chain of the probing period of one node

Figure 4. A node adaptively changes its probing period as time passes.

In a stable P2P system, most nodes stay at one place and only a limited number of nodes move occasionally. With the support of varying checking periods, nodes probe at the upper bound period thus a large portion of bandwidth is saved. While in a dynamic environment where many nodes join and leave the system from time to time, nodes probe their neighborhoods with a frequency proportional to the system dynamic degree.

3 Performance Evaluation

3.1 Simulation Methodology and Performance Metrics

We have evaluated SAT-Match comprehensively through simulation. The computing facilities are machines with four Intel Xeon 3.06GHz processors and 2GB memory running Linux 3.2.2-5.

The underlying physical topology is generated by the GT-ITM tool [13], which is used to generate Transit-Stub (TS) topologies in [8] and in [12]. GT-ITM generates two tiers topologies resembling the backbone and the edges of the Internet respectively. As far as the logical overlay is concerned, we build our own Java based CAN simulator by following the protocol described in [8]. Each peer on the overlay is uniquely mapped to one node in the IP layer. We choose CAN as the platform because it adapts well to dynamic environments. However, SAT-Match can also be easily deployed in other structured P2P systems such as Chord and Pastry.

We consider three metrics to verify the effectiveness of SAT-Match.

1) $R_s(t)$: *stretch reduction rate*. Denoting S_0 and $S_s(t)$ as the system that has a randomly mapped topology and the system with deployment of SAT-Match, respectively, we define *stretch reduction rate* at time t as follows:

$$R_s(t) = \frac{stretch(S_0) - stretch(S_s(t))}{stretch(S_0)}.$$

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

2) $R_{rt}(t)$: *response time (RT) reduction rate*, which has a similar definition as R_s:

$$R_{rt}(t) = \frac{RT(S_0) - RT(S_s(t))}{RT(S_0)}.$$

3) $O_c(t)$: *cumulative overhead*, which is defined as the accumulated overhead messages in bytes imposed to this system by SAT-Match at time t compared to the system without optimization.

There are several parameters that can be tuned in the simulation.

1) d: the dimensionality of the Cartesian space

2) k: the scale of the TTL-k neighborhood of one node

3) n: the size of the CAN system

4) *topology type*, which characterizes the node distribution on the transit domain and on leaf stub domains.

We have conducted experiments on the combination of d = {2, 3, 4}, k ={2, 3, 4}, n ={196, 1264, 4525, 5400}, and two different topologies respectively. In particular we focus on the large sized systems with 4525 nodes and 5400 nodes, which we denote as *ts5k-large* and *ts5k-small* respectively. The GT-ITM parameters specification of these two topologies are as follows.

- The *ts5k-large* is a large transit-stub topology where most nodes are on the leaf stub domains. Each transit node has 3 stub domains and there are 5 transit domains with a scale of 50. Transit domains have 5 nodes on average; with a probability of 0.6 there is an edge between each pair of nodes. For each stub domain on average there are 60 nodes in it, and the edge probability between any pair of nodes in this domain is 0.4.
- The *ts5k-small* is a small transit-stub topology where most nodes are in the transit domains. Each transit node has 4 stub domains and there are 120 transit domains with a scale of 50. Transit domains have 5 nodes on average; with a probability of 0.6 there is an edge between each pair of nodes. For each stub domain there are on average only 2 nodes in it.

We study the impact of each parameter on the above metrics.

3.2 Routing Hops Change

In the experiments we have used two different traces to study the change of routing hops caused by the deployment of SAT-Match, called REAL and PART. Both traces are used in [3]. REAL is a media streaming workload of 1663 clients; PART is a synthetic trace where object popularities satisfy the Zipf-like distribution. We have intensively measured routing hops distributions under different conditions before and after the topology adjustment. One representative result using the REAL trace is shown in Figure 5, where the mean routing hops on both curves are almost the same. We have observed that SAT-Match has almost no effect on the average routing hops.

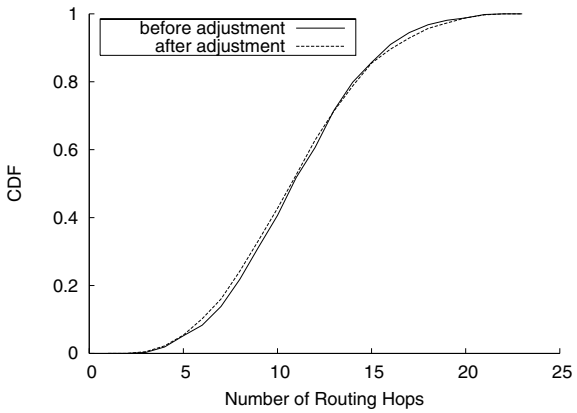


Figure 5. Routing hops distribution in a CAN of 1663 nodes with dimensionality of 4.

3.3 The Stretch Reduction

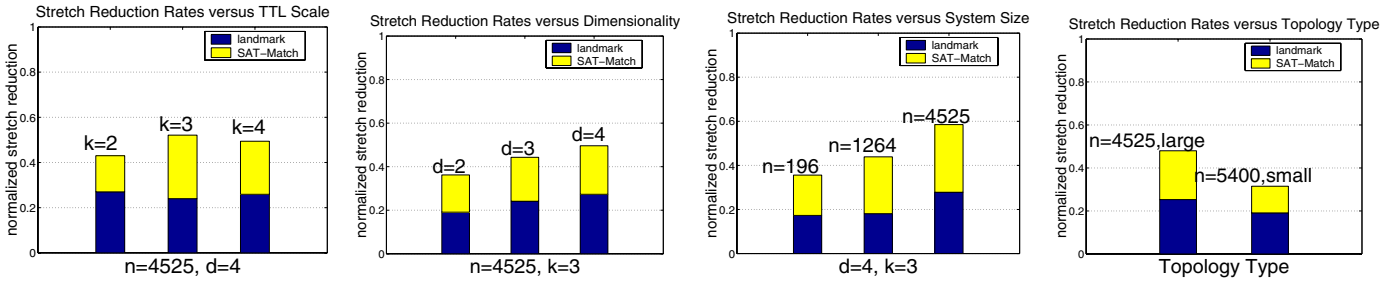
The *stretch* can be used to characterize the match degree of the overlay to the physical topology. With a fixed topology the stretch is equal to the average logical link latency normalized by the constant average physical link latency. Therefore we use the stretch reduction rate R_s as the primary metric in our evaluation. We show the impacts of those parameters on the stretch reduction rate in the following subsections.

3.3.1 A Comparison of Landmark Binning And SAT-Match

Figure 6 shows that even with the deployment of landmark binning technique, our adaptive adjusting scheme further reduces the system stretch about 27% when compared with the system without optimization. We have varied parameters such as TTL scale, dimensionality, system size and topology type and get similar results. In most cases the combination of these two techniques can achieve about 60% stretch reduction rate, among which 30% is due to the landmark binning technique, and the other 30% is due to the adatpvie adjusting and remapping scheme in SAT-Match.

This means that SAT-Match outperforms the landmark binning technique in terms of system topology match de-

0-7695-2132-0/04/$17.00 (C) 2004 IEEE



(a) Varying the TTL scale. (b) Varying the dimensionality. (c) Varying the system size. (d) Varying the topology type.

Figure 6. Landmark binning combined with SAT-Match.

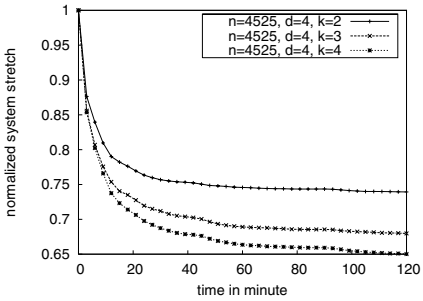


Figure 7. Varying the TTL scale.

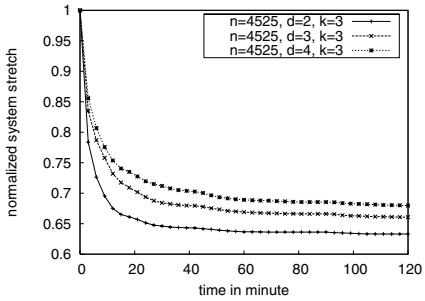


Figure 8. Varying the dimensionality.

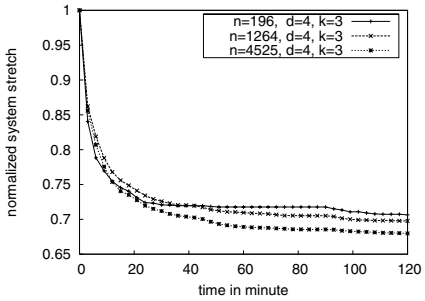


Figure 9. Varying the system size.

gree in most cases. In addition, they can be combined together to achieve significant stretch improvement compared to the system that no optimization is conducted.

3.3.2 The Impact of the TTL Neighborhood Scale k

Figure 7 illustrates the effect of the TTL scale k on the system stretch. We fix n to 4525 and d to 4, and vary k among $\{2, 3, 4\}$. In the beginning the stretch is reduced significantly and quickly because the mismatch degree is large and there is much room to improve. Then the reduction rate becomes smaller and smaller as time passes. After certain point of time the stretch can hardly be reduced due to structural constraints of the CAN. For the given scales of $\{2, 3, 4\}$, the corresponding stretch reduction rates after 100 minutes are 25%, 32%, and 35%, respectively.

On the other hand, although a larger scale can achieve a greater stretch reduction rate, it also causes heavier bandwidth consumption because of the increase in the probing scope. Considering the trade-off, we set k to 3 to achieve a high stretch reduction rate and to maintain the low overhead.

3.3.3 The Impact of Cartesian Space Dimensionality d

One observation in Figure 8 is that the increase in d results in the decrease of the stretch reduction. When d is 2, the stretch reduction rate is about 36% after 100 minutes. When d is 4, the stretch reduction rate is about 30% after 100 minutes. The gap is caused by the differences in the mismatch degrees between Cartesian space dimensionality and average node degree on the physical network. In the network topologies we have used, the average node degree is three. When d is 2, a node is most likely to have 4 logical neighbors. It slightly differs the average number of physical neighbors. When d is 3, on average a node has 8 logical neighbors; when d is 4, on average a node has 16 logical neighbors. The larger d is, the bigger the difference is between the logical overlay and the underlying physical topology, thus more difficult it is to adjust the overlay structure to reduce the stretch.

One benefit of increasing d is the significant decrease in the average number of routing hops, asymptotically denoted as $O(dn^{\frac{1}{d}})$. However, the increase in d can result in maintenance of more neighbor states in each node and can cause a significant increase in generated overhead traffic.

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

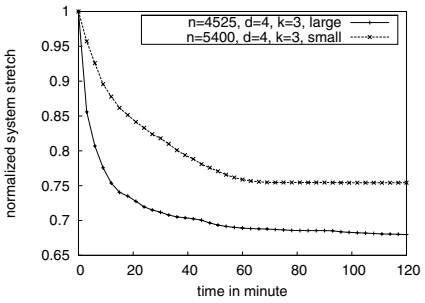


Figure 10. Varying the physical topology type.

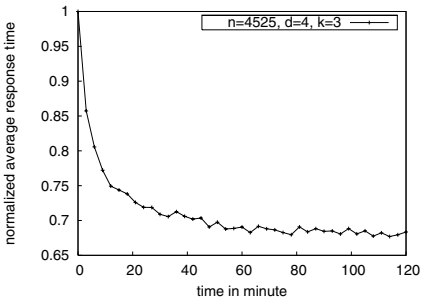


Figure 11. Average response time.

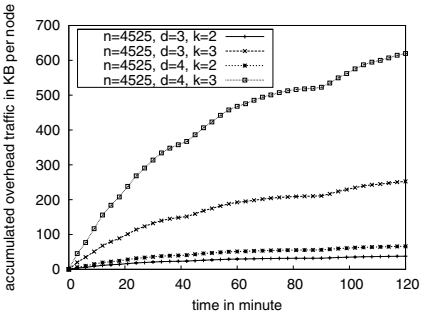


Figure 12. Cumulative traffic in KB.

3.3.4 The Impact of System Size n

Figure 9 shows that the system size n does not affect the stretch much. After 100 minutes, the 196 node system achieves a stretch reduction of 27%, and the 1264 node system achieves a stretch reduction of 30%, while the 4525 node system achieves the largest stretch reduction of 32%. However, the difference is below 5%. Since the systems have large sizes and close physical topology types, nodes have similar overlay connections and join and leave the system in a resembling way, thus the system stretch reduction is close.

3.3.5 The Impact of Physical Topology Types

As mentioned earlier, we have generated two different types of topologies *ts5k-large* and *ts5k-small* by the GT-ITM tool. Both topologies contain about 5000 nodes. *Ts5k-large* has more condense edge networks while *ts5k-small* has a larger backbone where a lot of nodes are attached.

Figure 10 compares the stretch reduction rates under these two topologies. It can be observed that *ts5k-small* outperforms *ts5k-large*. For the *ts5k-large* topology, the stretch reduction rate is only 12% after 100 minutes. While for the *ts5k-small* topology, the stretch reduction rate is 32%. which is 20% more There are more edges connecting transit domains in *ts5k-small* than in *ts5k-large*, thus its average node physical degree is larger, which contributes better performance gain.

3.4 The Average Response Time Reduction

Since as mention earlier, SAT-Match does not change the average routing hops, the reduction of single hop mean routing latency can result in the reduction of average routing response time. A smaller system stretch incurs a shorter response time.

In Figure 11, after SAT-Match is applied, the response time is reduced to 32% in merely 100 minutes. Again, SAT-Match can be combined with other approaches such as the landmark binning method or be implemented on top of eCAN to further reduce the response time.

3.5 The Imposed Overhead

We have done experiments by setting $d = \{3, 4\}$ and $k = \{2, 3\}$ for the purpose of comparing the overhead traffic of SAT-Match under different circumstances. From Figure 12 we can observe that the overhead increase rate drops as the time passes, which is controlled by our adaptive checking period mechanism. In the beginning a lot of jumps occur and most nodes probe frequently, thus the overhead increases drastically. As time elapses more and more nodes reach their proper positions, and probe their neighborhoods less frequently, which results in the curve dropping.

From Figure 12 we can also observe that when k is fixed, the bigger d is, the more overhead traffic is generated. The reason is that on average there are $(2d)^k$ nodes within the TTL-k neighborhood. The overhead traffic ratio of $\{k = 3, d = 3\}$ over $\{k = 3, d = 2\}$ is close to $156/64 = 2.43$.

On the other hand, when we fix the dimensionality d and change the TTL value k, the overhead traffic also changes significantly for the similar reason. Again the number of nodes inside any TTL-k neighborhood is roughly $(2d)^k$. When d is fixed and k increases, the overhead increases exponentially. For example, when $d = 2$ and $k = 2$, the size of a typical TTL-1 neighborhood is 16. While when $d = 2$ and $k = 3$, the size of a typical TTL-1 neighborhood is 64, which causes 4 times higher overhead traffic. Considering the trade-off between the stretch reduction rate and the overhead traffic, we have tuned the parameters in all ways and have found that $d = 4$ and $k = 3$ is an optimal choice.

4 Other Related Work

There are two methods that utilize local information to address the mismatching problem in P2P networks. The

0-7695-2132-0/04/$17.00 (C) 2004 IEEE

most related one to our SAT-Match is a system called Mithos [11]. The topology matching scheme is embedded into a multi-dimensional Cartesian space where each node is uniquely assigned a coordinate. By carefully selecting coordinate for each node and establishing quadrant links with neighboring nodes at the bootstrapping time, Mithos can improve the topology matching with local coordinate knowledge.

Originally designed for mesh alike overlay, Mithos has some limitations when being applied to a dynamic P2P environment to construct a topology-aware overlay. First, Mithos may not well handle frequent arrivals and departures of peers. When nodes leave, physically closest neighbor nodes may change, thus an adjustment mechanism is needed, which is not supported in Mithos. Second, Mithos may not well adapt to highly dynamic networking environment involving mobile nodes. Both the ID assignment, which uses virtual spring algorithm, and the quadrant links establishment algorithm in Mithos assumes the static physical networks. The self-adaptation in SAT-Match addresses the above two limitations. Finally, the overlay in Mithos is a multi-dimensional irregular mesh. This means that for each node, the number of forwarding neighbors is almost fixed. In a network where node degrees vary vastly, it is difficult for the overlay and the physical network to match well. In contrast, in our CAN based design of SAT-Match, the node degree is a variable, which makes it possible for a perfect match between the overlay and the physical network.

Another work being worth mentioning is an effective topology matching approach presented in [5],[6], which utilizes only local information to adaptively solve the mismatching problem. However, the methods are developed in the domain of unstructured P2P systems, and we can not directly use it in structured P2P systems.

5 Conclusion

This paper has described a novel adaptive scheme to match the P2P overlay to the physical topology by an iterative local optimization process from each node. Our intensive experiments have shown that with this scheme the stretch can be reduced up to 40%, while the overhead traffic generated is trivial. SAT-Match can be implemented without changing any structure of the existing CAN infrastructure and can be combined with many other topology-aware techniques such as landmark binning and eCAN.

Acknowledgment: We thank the anonymous referees for their constructive comments.

References

[1] M. Castro, P. Druschel, Y. Hu, and A. Rowstron. Exploiting network proximity in distributed hash tables. In *Proceedings of FuDiCo 2002*, Bertinoro, Italy, June 2002.

[2] Gnutella. http://www.gnutella.com/.

[3] L. Guo, S. Chen, S. Ren, X. Chen, and S. Jiang. Prop: a scalable and reliable p2p assisted proxy streaming system. In *Proceedings of ICDCS 2004*, Tokyo, Japan, March 2004.

[4] S. Jiang, L. Guo, and X. Zhang. Lighflood: an efficient flooding scheme for file search in unstructured peer-to-peer systems. In *Proceedings of ICPP 2003*, pages 149–160, Kaohsiung, Taiwan, October 2003.

[5] Y. Liu, X. Liu, L. Xiao, L. Ni, and X. Zhang. Location-aware topology matching in unstructured p2p systems. In *Proceedings of INFOCOM 2004*, Hong Kong, China, March 2004.

[6] Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni. A distributed approach to solving overlay mismatching problem. In *Proceedings of ICDCS 2004*, Tokyo, Japan, March 2004.

[7] Napster. http://www.napster.com/.

[8] S. Ratnasamy, P. Francis, M. Handley, and R. Karp. A scalable content-addressable network. In *Proceedings of SIGCOMM 2001*, pages 161–172, San Diego, CA, USA, August 2001.

[9] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of INFOCOM 2002*, New York, NY, USA, June 2002.

[10] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of SIGCOMM 2001*, pages 149–160, San Deigo, CA, USA, August 2001.

[11] M. Waldvogel and R. Rinaldi. Efficient topology-aware overlay network. In *Proceedings of HotNets-I*, Princeton, NJ, USA, October 2002.

[12] Z. Xu, C. Tang, and Z. Zhang. Building topology-aware overlays using global soft-state. In *Proceedings of ICDCS 2003*, pages 500–508, Providence, RI, USA, May 2003.

[13] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of INFOCOM 1996*, volume 2, pages 594–602, San Francisco, CA, USA, March 1996.

[14] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, UC Berkeley, April 2001.

0-7695-2132-0/04/$17.00 (C) 2004 IEEE