



Turning Heterogeneity into an Advantage in Overlay Routing

Zhichen Xu, Mallik Mahalingam, Magnus Karlsson
Internet Systems and Storage Laboratory
HP Laboratories Palo Alto
HPL-2002-126 (R.1)
July 12th, 2002*

E-mail: {zhichen, mmallik, karlsson}@hpl.hp.com

overlay
networks, peer-
to-peer, landmark
numbering,
AS topology

Recent P2P overlay systems, represented by Pastry, CAN and Chord, offer an administration-free and fault-tolerant application-level overlay network. While elegant from a theoretical perspective, these systems have some serious disadvantages. First, these systems rely on application-level routing, which may be inefficient with respect to network delays and bandwidth consumption. Second, they typically construct a homogeneously structured overlay and do not make discriminative use of nodes, since nodes in these networks usually have varying physical connectivity and packet-forwarding capacities.

In this paper, we propose two approaches for constructing an auxiliary network called an *expressway* to take advantage of the different connectivity and forwarding capacities of the nodes. As a result, we are able to reconcile the conflict of presenting the applications with a homogeneous structured overlay to simplify management, while at the same time taking advantage of the inherent heterogeneity of the underlying physical network. Our simulation results show that our expressway can achieve close to default IP routing performance (on average, 1.06 times default IP routing) in overlay networks.

Turning Heterogeneity into an Advantage in Overlay Routing

Zhichen Xu, Mallik Mahalingam and Magnus Karlsson
 Hewlett-Packard Laboratories
 1501 Page Mill Rd, Palo Alto, CA 94304, USA
 {zhichen, mmallik, karlsson}@hpl.hp.com

Abstract— Recent P2P overlay systems, represented by Pastry, CAN, and Chord, offer an administration-free and fault-tolerant application-level overlay network. While elegant from a theoretical perspective, these systems have some serious disadvantages. First, these systems rely on application-level routing, which may be inefficient with respect to network delays and bandwidth consumption. Second, they typically construct a homogeneously structured overlay and do not make discriminative use of nodes, since nodes in these networks usually have varying physical connectivity and packet-forwarding capacities.

In this paper, we propose two approaches for constructing an auxiliary network called an *expressway* to take advantage of the different connectivity and forwarding capacities of the nodes. As a result, we are able to reconcile the conflict of presenting the applications with a homogeneous structured overlay to simplify management, while at the same time taking advantage of the inherent heterogeneity of the underlying physical network. Our simulation results show that our expressway can achieve close to default IP routing performance (on average, 1.06 times default IP routing) in overlay networks.

Index terms — overlay network, peer-to-peer, landmark numbering.

I. INTRODUCTION

Providing scalable and efficient content storage and delivery are becoming more important because the demand for Internet-based applications is growing at an incredible rate. CDNs and network service providers [1, 2] advocate using overlays to provide scalable and robust Internet-based applications. These overlays are typically configured by administrators, but this is not feasible for large overlays because of the centralized nature of the construction process. Recent application-level overlay networks, such as CAN [3], Chord [4], and Pastry [5], are scalable, decentralized, and self-organizing. Nodes in these networks collectively contribute towards an administration-free storage space. The basic functionality these systems provide is a *distributed hash table* (DHT). Retrieving an object in these systems amounts to routing to the node that is responsible for storing that object. The routing path on these overlay networks is at the application level rather than at the IP level. Providing a simple and homogeneous abstraction has a number of desirable properties. It provides

stability by hiding the underlying dynamism and heterogeneity in the system, and simplifies the management of the system (e.g., by providing a uniform storage space to avoid hotspots) in a large-scale distributed environment.

While elegant from a theoretical perspective, these systems suffer from two limitations. First, they rely on application-level routing that largely ignores the characteristics of the underlying physical networks, which can lead to excessive routing delays. Second, they construct a homogeneous structured overlay network, while in reality, the nodes usually have different constraints such as load, packet-forwarding capacities, and network connections. In fact, for different nodes, the number of nodes that are physically close¹ to them can vary significantly. We support this claim by constructing an Autonomous System (AS) graph from BGP routing tables [6], which shows that the fan-out of an AS can range from 1 to 2621. We argue that for a system to function efficiently, it is important to make discriminative use of the nodes in the system. The question we try to answer in this paper is, whether we can represent heterogeneity of nodes without altering the homogeneous overlay network presented to the application.

In this paper, we describe two approaches for constructing an auxiliary network called an *expressway*, that takes physical proximity, forwarding capacity, and node connectivity into account, to significantly increase routing performance. The first approach uses the AS-level topology extracted from BGP reports. The second approach uses a novel *landmark numbering* strategy that can deal with the changing network conditions.

Earlier systems such as Pastry [5] and Tapestry [7] account for physical proximity when the overlay is constructed. As a result, the choices are limited to the nodes that are available at that moment. Topologically-Aware CAN [8] uses landmark

¹ By physically close, we refer to closeness in terms of network distance and we will use this term throughout the paper.

ordering to cluster nodes that are physically close into a logical vicinity during overlay construction. Though this yields a good performance improvement, it may cause significant imbalances in the distribution of the nodes in the CAN space that lead to hotspots [9] and does not handle changing network conditions. Chord, on the other hand, performs dynamic measurement for the nodes in the *finger table* and employs heuristics that selects the best next hop. However, the choices are limited to entries in the finger table. All the above systems, to a great extent, are constrained by the logical structure of the default overlay, possibly limiting the maximum performance that can be achieved.

Brocade [10] removes some of the constraints in previous systems by constructing a secondary overlay network of *supernodes* that are situated near the network access points such as routers. Though Brocade improves performance, it still uses logical routing in the secondary network, which incurs several physical hops for every logical hop. Brocade, to some extent, pushes the problem to an auxiliary network of a smaller size.

Our proposal decouples the homogeneous abstraction from routing altogether. It allows nodes to have a variable number of neighbors in an expressway, leaving the homogeneous structure of the default overlay network intact. In an expressway, nodes that are *well connected* and have good forwarding capacity establish connections with each other in a way that preserves physical proximity. These well-connected nodes collectively form an expressway. We use an expressway to propagate route information. To our knowledge, our expressway is the first unconstrained auxiliary network that takes full advantage of heterogeneity and the proximity in the underlying network.

The contributions of this paper are:

- Constructing an auxiliary network based on network proximity information using AS-level topology derived from BGP reports and a novel *landmark numbering* technique to cluster nodes that are physically close to each other and can handle changing network conditions.
- Route advertisement using a variation of the distance vector algorithm. In particular, we employ a route summarization technique to reduce routing state while trading routing

performance. This is equivalent to advertising network prefixes with the exception that the nodes that are aggregated are logically close to each other rather than physically close to each other.

- A simulation study using an Internet-like topology. To show the effectiveness of our techniques, we compare our approach with eCAN [13] and Brocade.

Our simulation results show that taking underlying physical network characteristics into consideration for routing in the overlay network yields significant performance improvement over the default algorithm. Our approach achieves close to default IP routing performance (on average 1.06 times default IP routing). In most cases, Brocade and eCAN stay within 3 to 4 times the performance of default IP routing. Brocade approaches the performance of default IP routing only when aggressive caching (caching address advertisements everywhere) is used. Even then, we would expect expressway routing to outperform Brocade in terms of multicast performance because expressway closely approximates the underlying physical network topology. In fact, constructing an overlay that closely approximates the physical network makes it possible to deliver routing performance that is better than default IP routing. This has been shown by RON [11] and Detour [12].

The remainder of the paper is organized as follows. Section II provides background on CAN and eCAN that we use in our study. Section III describes different ways to construct expressways and their protocols. In Section IV we evaluate the approaches using simulation, and present our results. Section V provides related work, and we conclude the paper in Section VI.

II. DEFAULT OVERLAY NETWORKS

The approaches that we have proposed in this paper are generic and applicable to many overlay systems such as CAN, Pastry and Chord. We evaluate our approaches against Brocade [10] and eCAN [13]. eCAN is a topology-aware version of CAN that delivers logarithmic logical routing performance and is independent of the dimension. We briefly describe CAN and eCAN below.

A. CAN and eCAN

CAN abstracts the problem of data placement and retrieval over large-scale storage systems as hashing that maps “keys” onto “values”. It organizes the logical space as a d -dimensional Cartesian space (a d -torus). The Cartesian space is partitioned into zones, with one or more nodes serving as owner(s) of the zone. An object key is a point in the space, and the node owns the zone that contains the point stores the object. Routing from a source node to a destination node boils down to routing from one zone to another in the Cartesian space. Node addition corresponds to picking a random point in the Cartesian space, routing to the zone that contains the point, and splitting the zone with its current owner(s). Node removal amounts to having the owner(s) of one of the neighboring zones take over the zone owned by the departing node. In CAN, two zones are neighbors if they overlap in all but one dimension along which they neighbor each other.

eCAN augments CAN’s routing capacity with routing tables of larger span. Every k CAN zones represent an order-2 zone, and k order- i zones represents an order- $i+1$ zone. The variable k is called the zone coverage factor of eCAN. It can be shown that setting k to e^d achieves the optimal logical routing performance, where d is the dimension of the default CAN. As a result, a node is an owner of a CAN zone and is also a resident of the high-order zones that encompass the CAN zone. Besides its default routing neighbors that are CAN zones, a node also has *high-order routing neighbors* that are representatives of its neighbors in the high-order zones. eCAN provides flexibility in selecting the high-order neighbors. When selecting representatives for a high-order neighbor, we can select the node that is closest to the current node among all the nodes that belong to the neighboring high-order zone.

Figure 1 illustrates eCAN with an example. The default CAN zones are order-1, and each of the CAN zones is $1/64$ of the entire Cartesian space. In this example, four neighboring CAN zones make one order-2 eCAN zone and four order-2 zones make an order-3 zone. For example, node 1 owns a CAN zone (the zone with dark shading in the upper-left corner), and it is also a resident of the order-2 and order-3 eCAN zones that enclose the CAN zone. The routing table of node 1 consists of the default

routing table of CAN (represented by the thin arrows) that link only to node 1’s immediate CAN neighbors, and the high-order routing tables (represented by the thick arrows) that link to one node in each of node 1’s neighboring eCAN zones at order-2 and order-3. Figure 1 also shows how node 1 can reach node 9 using eCAN routing (1 – 2 – 5 – 9).

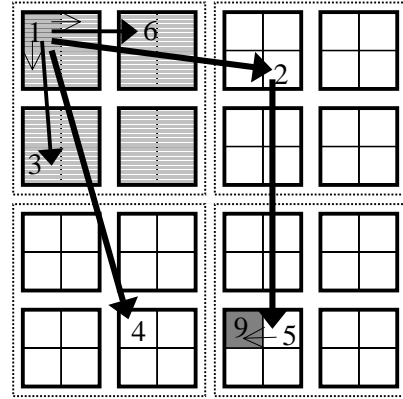


Figure 1: An example of eCAN routing

III. TOPOLOGY-AWARE EXPRESSWAY

In an expressway, each node establishes connections with nodes in its physical proximity that are *well connected* and have good forwarding capacities. These well-connected nodes are called *expressway nodes*. The expressway nodes themselves are linked to other expressway nodes that are close by, called *expressway neighbors*, to form an expressway. We use *distance-vector-based* route advertisement in an expressway and employ a summarization technique to reduce the routing state. In our work, expressway nodes serve three purposes: (i) to propagate routing information when nodes join or leave, or the network condition changes; (ii) to resolve the routing destinations; (iii) to forward packets for multicasting or for better IP routing performance.

In the subsections that follow, we describe how expressway neighbors are selected, how routing advertisement among expressway nodes is performed, and how the routing algorithm works.

A. Selection of expressway neighbors

We use two different approaches for selecting expressway neighbors. The first approach is based on deriving AS-level topology information and

clustering nodes that are part of the same AS. While this information is fairly simple to derive, there are a few issues with respect to the way the AS-level graph is used for selecting expressway neighbors. First, the distance information provided by the AS graph can be too coarse grained. Second, AS graph captures only static information. Third, there can be many AS that do not participate in the overlay. Consequently, an expressway may not be fully connected. To address these problems, we propose an alternative technique, which is based on clustering nodes that observe the same latency behavior with a few chosen landmarks. We describe the details of these two approaches in the following sections.

1) Expressway using AS-level topology

When a node joins an expressway, it determines the AS to which it belongs (by mapping an IP address to an AS-ID using BGP reports), and publishes information that includes its IP address, its logical ID (e.g., node ID in the case of Chord and Tapestry, and the CAN zone a node owns in the case of CAN), and whether it is an expressway node. This published information is kept in the overlay system itself using a hash of the publishing node's AS-ID as the key. The global state published on the default overlay does not need to be kept consistent because it will not affect the correctness of the routing.

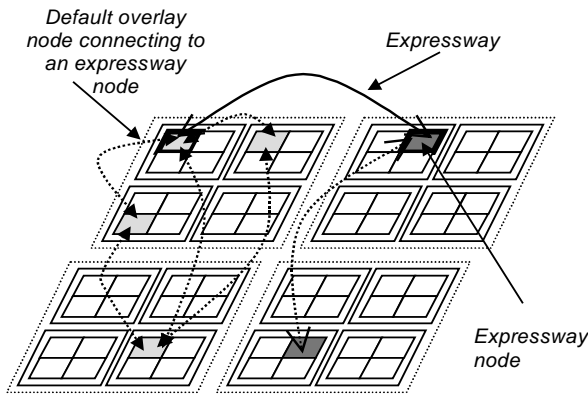


Figure 2: Selection of expressway neighbors using AS topology

An *ordinary node* (a non-expressway node) uses the ID of its AS to get the information on all other nodes in the same AS, and establishes direct connection with the nodes in the same AS by adding them as its expressway neighbors. It notifies its

expressway neighbors to become part of their expressway neighbor set as well. In practice, this can be done through a caching scheme. All ordinary nodes find the expressway node that is closest to them, and establish a connection with it. The first time an ordinary node tries to communicate with a local ordinary node, it will always go through the expressway node and cache the address of the destination. It will communicate to the destination directly from then on.

If the new node is an expressway node, it also uses the IDs of its neighboring AS (that it obtains from the AS-level topology) as hash keys, to obtain lists of *expressway nodes* in the overlay that are its topological neighbors and to establish direct connection with them. When a node joins or leaves the system, its expressway neighbors need to be notified and the default overlay needs to be maintained.

Figure 2 illustrates how expressway neighbors are selected, using eCAN as an example. The nodes that are in the light gray zones belong to the same AS and the nodes that are in the dark gray zones belong to a neighboring AS. The nodes with thick borders are expressway nodes. The expressway nodes establish connections between themselves to form an expressway. Ordinary nodes establish connections with expressway nodes in their proximity. In addition, each node in a region (AS) establishes connections with other nodes in its region.

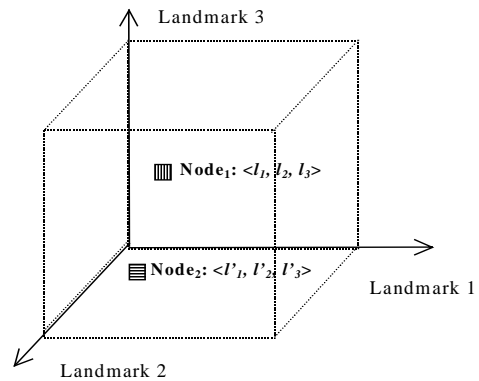


Figure 3: Landmark coordinate space for 3 landmark nodes

2) Expressway using landmark clustering

We pick n landmark nodes that are randomly scattered in the Internet. These landmark nodes can

be part of the overlay itself or standalone. Each expressway node measures its latencies to the n landmarks. For node A, suppose that the measured distances are $\langle l_1, l_2, \dots, l_n \rangle$. We then position node A in an n -dimension Cartesian space using $\langle l_1, l_2, \dots, l_n \rangle$ as its coordinates. We call this Cartesian space, the *landmark space*. The intuition behind doing this is that nodes that are close to each other have similar landmark measurements, and are close to each other in the landmark space². Figure 3, shows an example for a coordinate space using 3 landmarks.

The expressway nodes independently determine their positions in the landmark space and publish their positions on the default overlay. An ordinary or an expressway node finds expressway nodes that are physically close to it by referring to this published information.

The difficulty in storing the position information in landmark space of the expressway nodes is that the landmark space is of relatively high dimension, whereas the overlay itself is of relatively low dimension (e.g., 2). To solve this problem, several techniques can be used: using space-filling curves, extending the notion of *multiple reality* introduced by CAN, and employing nonlinear dimensionality reduction techniques such as locally linear embedding [14]. We elaborate the first two approaches below.

a) *Using space-filling curve*

Space-filling curves map points in the domain \mathcal{R}^1 (the domain of real numbers) into \mathcal{R}^d (a d -dimension Cartesian space) so that the closeness relationship among the points is preserved. If two points are close to each other in \mathcal{R}^1 , they will also be close to each other in \mathcal{R}^d . One example of space-filling curves is the Hilbert Curve [15]. The Hilbert curve is defined recursively. For an approximation level equal to 1 it is a point. For an approximation level equal to 3, it looks similar to Figure 4b. For each higher approximation level, we subdivide the entire space into four sub-zones and

copy a shrunken and possibly rotated version of the current approximation into each sub-zone.

We partition the landmark space into 2^{nx} grids of equal size (where n refers to number of landmarks and x controls the number of grids used to partition the landmark space), and number each expressway node according to the grid into which it falls. We call this number the *landmark number* of the node. Closeness in landmark number indicates physical closeness. The smaller the x , the larger the likelihood that two nodes will have the same numbering, and the finer grain the physical proximity information.

For CAN, we can partition the overlay into grids, and store the information about expressway nodes in a grid depending on its landmark numbering, again using a space-filling curve (see Figure 4). We can employ a similar procedure for other overlay networks. For example, in the case of Chord, we can simply use the landmark number as the key to store the information of an expressway node on a node whose ID is equal to or greater than the landmark number. In the case of Tapestry, we can use a prefix of the node IDs to partition the logical space into grids. In summary, our goal is to store expressway node information such that information about close-by nodes is stored close to each other in the overlay.

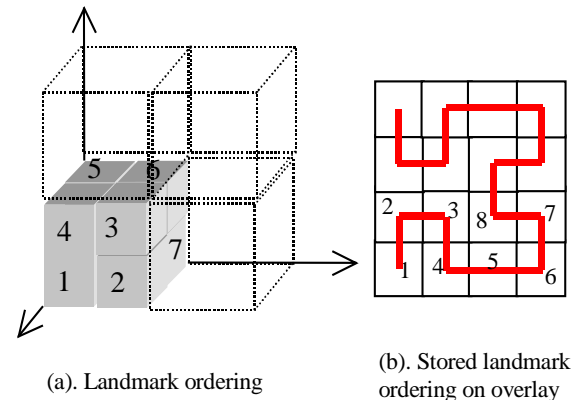


Figure 4: Mapping a 3-dimensional landmark space to 2 dimensions using the Hilbert curve

b) *Using multiple reality*

Alternatively, we can construct a high-dimension CAN just for the purpose of storing the positions of the expressway nodes in the landmark space. Because the number of expressway nodes in the system can be much smaller than the total number of

² A sufficient number of landmarks need to be chosen to reduce the probability of false clustering where nodes that are far away in network distance are clustered close to each other.

nodes in the system, we can select only a small number of nodes for the high-dimension CAN space. These nodes have multiple realities: the default overlay, which can be Pastry, Tapestry, Chord, and other types of overlay; and the CAN space for storing landmark information.

In the rest of the paper, we assume space-filling curves are used. The ordinary nodes locate the expressway node that is *closest* to it by (i) computing its own landmark number; (ii) using its landmark number as the key to route to the location (destination location) where information about other nodes that have similar landmark numbers is stored; and (iii) performing a *controlled flooding* starting from the destination location until information for some expressway node is encountered. Controlled flooding is possible because information about close-by nodes is stored close to each other on the overlay.

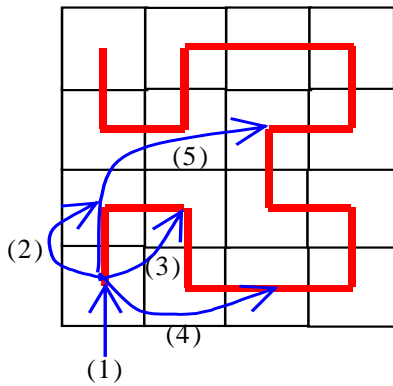


Figure 5: Expressway construction using landmark clustering. (1) Node N with landmark number 0 gets information about other nodes with the same landmark number and establishes a connection with any one of them; (2) Node N gets information about nodes with landmark number $1(2^0)$; (3) Node N gets information about nodes with landmark number $2(2^1)$; (4) Node N gets information about nodes with landmark number $4(2^2)$; (5) Node N gets information about nodes with landmark number $8(2^3)$.

We describe how to build a fully connected expressway using the landmark number stored in the default overlay itself. The basic idea is to have each node connect to a number of other nodes whose landmark numbers are numerically close to its landmark number. This approach is completely decentralized. Figure 5 shows how the nodes with landmark number 0 select their expressway

neighbors. The neighbors are selected based on their landmark numbers that are at 2^i distance from the current node's landmark number.

An expressway could be constructed in many different ways. One way is to subdivide the landmark space into grids and higher-order grid spaces as in eCAN. Each node will elect leaders for each of the grid spaces using a predetermined selection algorithm. These leaders then form the high-order grid. Another possibility is to construct a proximity-aware overlay such as in eCAN (or the topologically-aware CAN [8]), to take advantage of the self-organizing feature of a P2P overlay.

One important goal is to minimize the cost of maintaining the expressway connections. As expressway nodes themselves can join and leave the system, it is important to build some redundancy into the expressway system. It should be noted that whether the expressway is fully connected or not affects only routing performance not the correctness of the routing algorithm. In summary, using landmark numbering ensures that the expressway has good physical proximity. This can reduce the latency in propagating the route advertisements and expressway routing itself. Landmark numbering can be performed repeatedly to reflect the changing network conditions.

B. Route advertisement with summarization

When a node joins an expressway, it advertises all the local nodes that are in its physical proximity to its neighboring expressway nodes. Instead of advertising the logical IDs of the nodes directly, we employ a summarization scheme to provide a means to control the amount of routing state maintained at each node (to trade-off between routing performance and routing table size). For CAN, the summarization is based on the notion of virtual grids in the Cartesian space and a numbering scheme. For other DHT-based overlays such as Pastry or Tapestry, the prefix or suffix of the nodes can be used to summarize the logical space to achieve the same effect.

For CAN, the entire d -dimensional Cartesian space is partitioned into m^d virtual grids of equal size, and each virtual grid is assigned a number ranging from 0 to $m^d - 1$, called the virtual grid ID. Each default CAN zone is summarized using the ID

of the virtual grid in which the center of the CAN zone falls. Similarly, any point in the Cartesian space is numbered using the same scheme.

Figure 6 illustrates the partitioning of a 2-dimensional Cartesian space using 4^2 grids. A node uses the ID of the grid to which it maps during route advertisement. For example, if a node has a coordinate of $\{0.1, 0.3\}$, then it uses virtual grid id of 4 during route advertisement.

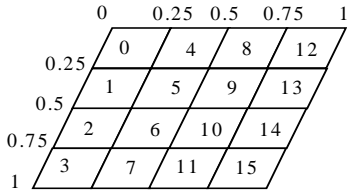


Figure 6: Summarizing a 2-dimensional Cartesian space using 4^2 grids

The algorithm for route advertisement is the same as the standard distance vector algorithm. The differences are that (i) instead of advertising just a node's transport address, we also advertise the virtual grid ID that is used for summarizing nodes; (ii) only expressway nodes participate in route advertisement; (iii) the route advertisement messages are controlled with a time-to-live (TTL) value expressed as a number of expressway-node hops. Having a small number of virtual grids would produce less precise advertised information but the route state that an expressway node needs to maintain becomes smaller. Even when the virtual grid is larger than the zone to be advertised, routing to any zone that belongs to the virtual grid guarantees that the target is inside the virtual grid and can be routed with the default overlay routing in a bounded number of logical hops.

In addition to the default routing table for eCAN, each ordinary node keeps the addresses of the local expressway nodes, whereas expressway nodes maintain route summaries. The number of entries in a route summary is on the order of the number of virtual grids used for summarizing the nodes in the system.

Expressway routing can be done in two different ways: using the IP address propagated in route advertisement (direct route), and using the expressway nodes to forward the packets (expressway-node forwarding). The first approach

requires slightly more storage space to keep the route summary and relies on IP routing. However, if a node leaves the network, the second approach is less expensive to repair because the grids to summarize the zones does not change even when nodes join and leave the system. In addition, forwarding on the expressway provides a way to handle multicast efficiently, and possibly deliver performance better than default IP routing.

C. Routing Algorithm

Routing with an expressway in place is fairly straightforward. When an expressway node receives a packet it performs the following operations.

1. It checks the route summary. If the destination is in the route summary, then it routes the packet to the destination directly.
2. Otherwise, it checks to see if there is any expressway node that is logically closer³ to the destination than the current node. If it finds such a node, it tags the packet and then forwards the packet to that expressway node.
3. Otherwise, it tags the packet and uses the default routing (i.e., eCAN).

For non-expressway nodes, the algorithm is slightly different.

1. If the packet is tagged, it uses default routing.
2. Otherwise, it checks to see if the destination is one of its eCAN neighbors. If so, it routes the packet to that neighbor directly.
3. Otherwise, it sends the packet to one of the local expressway nodes.

If the destination is in the route summary, the expressway will route the packet to the destination or to a node that is logically close to the destination. If the expressway routes the packet to a node that is not the destination, from then on the packet is tagged and only default routing will be used. If it is not in the route summary, the packet will be tagged, and only default routing will be used. This guarantees that the packet will reach the destination.

³ If the expressway is fully connected then we can send the packets to an expressway node that is physically closest to the destination.

D. Dynamism and Scalability Issues

It is reasonable to expect dynamism in the node membership in a large-scale widely distributed environment. This significantly impacts the routing state maintained by a node. However, we believe that events such as joining and leaving are less frequent in the case of expressway nodes. For a highly dynamic environment, using an auxiliary network is probably not a best choice. Even caching cannot help in that case and the system will have to rely on the default overlay routing.

As far as route propagation is concerned, the nodes that reside in the virtual grids can change dynamically. How timely this information can be propagated is an open issue. However, we believe it is not critical. An expressway will perform at least as good as Brocade even if we decide not to perform route update when an owner of a virtual changes. In this case, the source node sends the packet directly to the expressway node that has information about the target virtual grid. This is in essence, how Brocade works.

In essence, we have used the default overlay as a rendezvous plane where the expressway nodes publish information about themselves and all the nodes retrieves that information. This is done in a totally decentralized fashion without any centralized coordination. If a node that hosts this information leaves the system, we leverage the mechanisms provided by the underlying overlay. In addition, we can also refresh this information periodically. This is not an issue because periodically refreshing the global information is necessary to keep up with the changing network conditions.

IV. EVALUATION

A. Simulation Methodology

We use simulation to understand the performance gains of our approaches. We compare expressway against several different approaches: default IP routing, a variation of eCAN, and few variations of Brocade that use eCAN as the basis instead of Tapestry.

Rather than using a naïve eCAN that ignores the different forwarding capacity of nodes, the variation of eCAN we compare against employs a simple strategy such that only expressway nodes participate in high-order routing. In this way, we evaluate our

new approach against solutions that are comparable but work in the framework of a default overlay.

We accomplish this by having only expressway nodes publish their IDs and locations on the overlay itself. In eCAN, each node independently computes its landmark number against a number of landmark nodes and publishes its ID on the overlay. A node finds its high-order neighbors by finding nodes in the appropriate neighboring high-order zones that are physically closest to it using its own landmark number as a reference. To make the comparison fair, the nodes in eCAN keep roughly the same amount of routing state as the new approaches. The main difference is that in the new approaches only the expressway nodes keep the extra routing state.

We use eCAN as the basis for Brocade because we want to compare against Brocade in the same setting, i.e., to compare against the idea of using a logical overlay as an auxiliary network for advertising the addresses of the ordinary nodes. In addition, we do not have access to a Brocade simulator. We believe the comparison is fair because both eCAN and Tapestry achieve logarithmic logical routing performance and are topology-aware.

For Brocade, all nodes besides the expressway nodes (*supernodes*) advertise their IDs and locations in the auxiliary network using summaries of their zones as keys to place the advertisement in the overlay. We simulated four versions: no caching of advertisements, caching along the path of advertisements, caching along both advertisement and retrieving paths, and caching advertisements everywhere in the auxiliary network. To make the comparisons fair, all Brocade simulations advertise the IP addresses of the nodes such that once the advertisement is located, direct IP routing will be used. In addition, we also assume nodes know the IP addresses of all other nodes in the same AS, which is the assumption made by Brocade.

The primary performance metric that we use is *physical routing delay* incurred in resolving a point in the Cartesian space from a randomly chosen node in the default overlay. Average routing delay is computed by taking a number of samples equal to ten times the number of nodes in the system (random nodes and random points in the default overlay).

We derive the delay of each resolution from the actual number of AS hops taken and the average delay between two neighboring AS. We have generated an approximate topology of the Internet by extracting AS connectivity information from BGP reports and computing shortest path distances between all pairs of AS. In our experiments we use 1,000 AS from a total of 13,000 active AS. When nodes in the default overlay are populated, each node is assigned to one of the 1,000 AS.

We assume an average inter-AS delay of 100 ms and an average intra-AS delay of 1 ms in the experiments. Although we would like to introduce some variance into inter-AS delay it is hard to do this in a consistent manner, as we only use AS hops not the complete AS path. Introducing variation consistently would require us to set the delays for all edges of a complete AS graph. In addition, for the size of AS graph we use, small number of landmark nodes seems to be sufficient.

Parameters	Default	Range
# of nodes (n)	-	512 - 8K
TTL	6	1 - 6
Virtual Grid	\sqrt{n} ^{*4}	$\sqrt{n} - \sqrt{n}/4$
# of Landmarks	5	-
Routing	Direct	direct, forwarding

Table 1: Parameters used in the simulation

In our experiment, we ignore the overhead in processing the packets, as we do not have a real implementation yet. We have implemented expressway construction using a leader selection algorithm for evaluation purposes.

The parameters that are varied in our experiments are the number of nodes, the size of the virtual grid, and the TTL. Table 1 summarizes the parameters and their values used in the simulation

B. Results

Figure 7 shows the effect of varying the number of nodes in the system from 512 to 8K. We fix the TTL, the number of virtual grids, and the number of landmarks using the default values shown in Table 1.

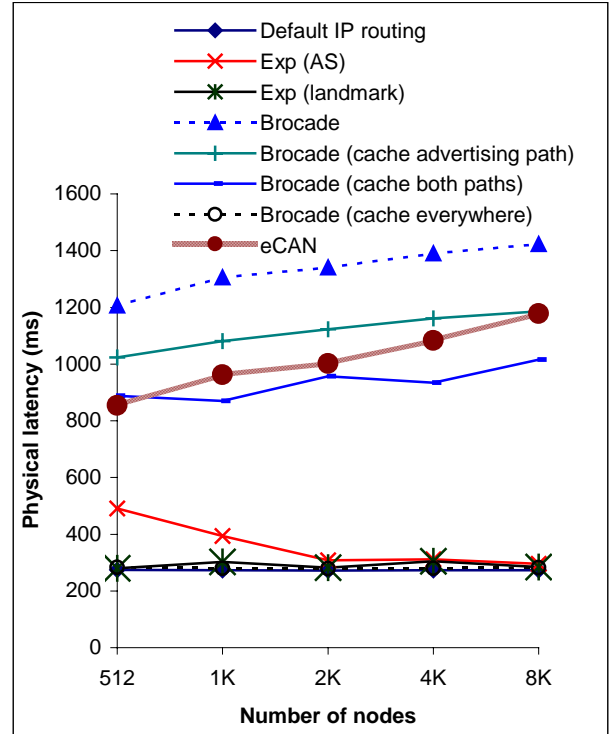


Figure 7: Comparison of various approaches

The curve labeled IP shows the average routing delay when default IP routing is used. The curve labeled eCAN corresponds to a version of eCAN that keeps approximately the same amount of routing state as our new approaches. The curve labeled Exp (AS) constructs the expressway using AS-level topology information, whereas the curve labeled Exp (landmark) employs the landmark-clustering scheme to construct a fully connected expressway in addition to applying AS neighboring information. The curves labeled Brocade show the performance of Brocade with different caching strategies.

From Figure 7 we observe the following.

1. Constructing an expressway by using AS neighboring information can produce good routing performance. This is true especially when the AS graph is well populated (the number of nodes is greater than or equal to 2K in our experiments). On average the routing performance is about 1.3 times default IP routing with individual measurements ranging from 1.08 to 1.79 times default IP routing.
2. Expressway routing using landmark clustering produces close to optimal routing performance. On average the routing performance is about

⁴ It is actually the smallest power of 2 numbers that is greater than or equal to \sqrt{n} .

1.06 times default IP routing with individual measurements ranging from 1.02 to 1.11.

3. eCAN with similar total routing state performs better than a naïve Brocade implementation and Brocade that caches the routing advertisement only along the publishing path in the auxiliary network. Its performance is comparable to Brocade that caches also along the retrieving paths.
4. Brocade with aggressive caching (caching the route advertisements everywhere) obtains similar performance to default IP routing and expressway routing using landmark clustering. The key difference comes from the fact that the expressway closely approximates the physical network, whereas Brocade is a logical overlay. As a result, using expressway for multicast would be more efficient than using Brocade. In addition, when the auxiliary network closely approximates the underlying physical network, it would be more efficient to propagate the routing state along the physical overlay than along a logical overlay.

In our experiments, the default IP routing performance is independent of the number of nodes in the system. This is because all experiments use the same set of AS and the nodes in the overlay are randomly populated in the AS graph. By choosing the same set of AS and populating the topology with a variable number of nodes, we show how the node density can affect routing performance. For Exp (AS), increased node density implies a greater likelihood that the expressway is fully connected.

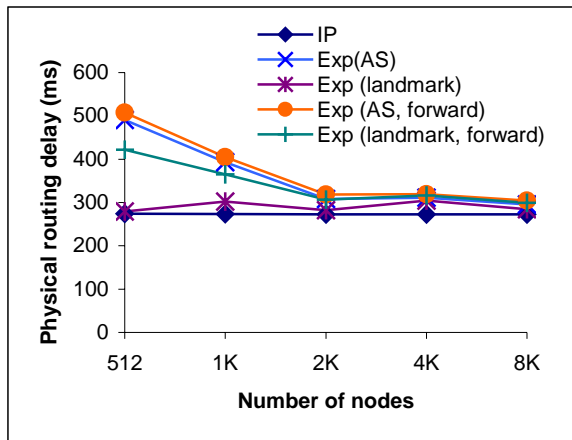


Figure 8: Comparisons of direct route and expressway node forwarding

Figure 8 compares direct routing with expressway-node forwarding. From the figure, we can see expressway-node forwarding provides reasonably good performance. The performance improves with more nodes in the system. This is because our implementation of landmark clustering is relatively naïve (e.g., the expressway is relative flat), and when the expressway becomes better connected because there are more nodes, the distance vector algorithm is able to find better routing paths. In theory, the performance of our expressway should approach that of RON [11], which performs better than default IP routing.

To give a feel for the cost paid to achieve near optimal performance, Table 2 shows the routing state that each node needs to keep for expressway routing using landmark clustering.

# of Nodes	# of Grids	Routing table (# entries)	
		Expressway (Avg)	eCAN (Avg)
512	32x32	1K	43
1024	32x32	1K	45
	16x16	256	46
	8x8	64	46
2048	64x64	4K	49
4096	64x64	4K	53
	32x32	1K	52
	16x16	256	52
8192	128x128	16K	57

Table 2: A glimpse of the routing state

Note that only expressway nodes need to maintain the route summary. With modern computer systems, we believe keeping this amount of state should not be an issue. In our experiments, we assume that each AS has at least one expressway node. This can vary in an actual implementation. If there is no expressway node in an AS, we can always rely on landmark numbering to locate an expressway node in an AS that is close-by.

The trade-off is that when there are more expressway nodes the non-expressway nodes need to keep less routing state. The state each expressway node needs to keep depends on the number of nodes in the system and the number of virtual grids used. However, the number of packets each expressway node needs to handle depends only on the number of

expressway nodes in the system. As the number of expressway nodes decreases the load on them increases.

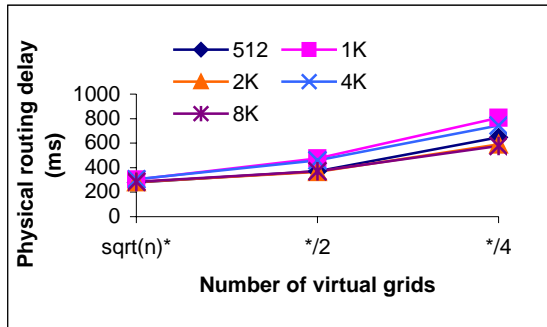


Figure 9: Varying the number of virtual grids. \sqrt{n}^* represents the smallest power of 2 number that is greater than or equal to \sqrt{n} , and n is the number of nodes

To understand the effect of varying different parameters that affect routing performance, we first show the results for varying the number of virtual grids used in a route summary. We use default values for TTL and landmarks. The results, which are shown in Figure 9, show that reducing the number of grids can significantly affect routing performance. However, the effect is only proportional to the square root of the number of grids. This indicates that we can obtain a relatively bigger saving in routing state with a relatively smaller sacrifice in performance.

In our experiments, when the virtual grid is occupied by multiple nodes, we assume that default overlay routing will be used once the packet reaches the node that represents the destination virtual grid. To address this problem a direct route can be used by, for example, employing some kind of local gossiping protocol for the nodes in the same virtual grid to exchange their IP addresses. Consequently, the performances will be at most twice as much as default IP routing.

Figure 10 shows the results for varying the TTL parameter. We use default values for landmarks and virtual grids. It can be seen that the TTL value can significantly affect routing performance if not chosen correctly. In certain cases, increasing the TTL by one can almost cut routing performance by half. In our experiment, a TTL of 4 is enough to bring the routing performance to close to optimal.

Other parameter that affects the routing performance is the number of landmarks used for generating landmark numbering. We did not see a significant difference in the routing performance when the number of landmarks is varied.

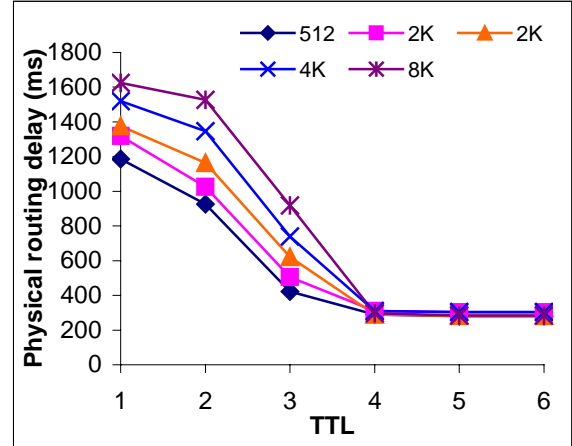


Figure 10: Effect of varying TTL for route advertisement

V. RELATED WORK

In P2P overlay networks that are based on Plaxton's proposal such as Pastry [5] and Tapestry [7], routing table entries point to near-by nodes when the routing table is constructed. However, the nodes that are examined represent only a small set of candidates. In addition, Plaxton's proposal was originally intended for a centrally administered environment: Web caching. Nodes in such systems cannot independently discover their neighbors in a decentralized fashion [3]. For instance, in Pastry, a new node X first has to contact a node A that is physically close to it. It asks node A to route to node X 's ID. Along the way, the routing table for X is constructed by picking up the level i routing table from the i^{th} node encountered.

In topologically-aware CAN [8], when a new node joins the overlay, it joins a node that is close to it in IP distance. This can result in uneven node distribution and holes in the Cartesian space, which can create overhead in data placement. Chord [4], in contrast, does not take the physical topology into consideration when constructing the overlay. Instead, it employs heuristics that use many logical hops with small latency instead of fewer logical hops with large latency. However, the choices are limited to entries present in the routing tables.

In Brocade [10], a secondary overlay network of *supernodes* is used to improve the routing distance. These supernodes are the nodes that are situated near the network access points such as routers and gateways. Nodes in the default network establish direct connections with a supernode that is close by. Supernodes advertise the nodes that are connected to them as *objects* served by the supernode in the secondary overlay. Routing from node A to D in the default network involves three steps: locating a supernode B locally, routing to the supernode C that stores the object D in the secondary overlay, and hopping from C to D.

Though Brocade produces some reasonable improvements from the default case, it pushes the problem to an auxiliary network of a smaller size. Brocade faces a dilemma in choosing the appropriate number of supernodes: if the number of supernodes is large, the logical overlay routing cost gets higher; if the number of nodes in the secondary network is small, an ordinary node needs to keep more state about the addresses of other nodes that are linked to the same supernode. This can be an issue in a dynamic environment. An expressway closely approximates the physical network, making performance independent of the size of the network.

In our simulation study, Brocade approaches default IP routing performance only when aggressive caching is used. Even then, we expect expressway routing to outperform Brocade with respect to multicast performance. In addition, expressway opens up the possibility of delivering routing performance better than default IP routing.

VI. CONCLUSION

In this paper, we described generic techniques to construct an auxiliary network for any DHT-based overlay to take advantage of the inherent heterogeneity that exists in the physical network.

Our approaches use proximity information for constructing an auxiliary network called an expressway. We use two different techniques to derive proximity information. The first approach uses an AS-level topology that is derived by processing BGP reports. The second approach uses a novel landmark clustering technique that clusters nodes that have similar latency behavior. As a result, we are able to reconcile the conflict of presenting

the applications with a homogeneous structured overlay to simplify management, while at the same time exploiting the inherent heterogeneity of the underlying physical network.

We conducted a detailed simulation study using a real Internet topology. The simulation results show that our approaches can achieve close to default IP routing performance. Our expressway based on landmark-clustering approach yields 1.06 times default IP routing performance on average, while AS-level-topology approach yields about 1.3 times default IP routing performance.

Although, we have evaluated expressway using eCAN as the default overlay, the techniques proposed in this paper are generic and directly applicable to other DHT-based systems as well.

In the future, we plan to do a detailed simulation incorporating fine-grained topology information and use other topology simulators such as GT-ITM [16] and BRITE [17] for expressway routing.

VII. ACKNOWLEDGEMENTS

We would like to thank Ira Greenberg and Chunqiang Tang for their valuable feedback on this paper.

VIII. REFERENCES

1. Inktomi, *The Inktomi Overlay Solution for Streaming Media Broadcasts*, http://www.inktomi.com/pdfs/whitepapers/overla_ywhtpapr.pdf.
2. Internap, <http://www.internap.com>.
3. Ratnasamy, S., et al. *A Scalable Content-Addressable Network*. in *ACM SIGCOMM*. 2001. San Diego, CA, USA.
4. Stoica, I., et al. *Chord: A scalable peer-to-peer lookup service for Internet applications*. in *ACM SIGCOMM*. 2001. San Diego, CA, USA.
5. Rowstron, A. and P. Druschel. *Pastry: Scalable, distributed object location and routing for largescale peer-to-peer systems*. in *IFIP/ACM Middleware*. 2001. Heidelberg, Germany.
6. Telstra, *Raw BGP Data*, <http://kahuna.telstra.net/bgp2>.
7. Zhao, B.Y., J.D. Kubiatowicz, and A.D. Joseph, *Tapestry: An infrastructure for fault-resilient wide-area location and routing*. 2001: Berkeley, CA, USA.

8. Ratnasamy, S., et al. *Topologically-Aware Overlay Construction and Server Selection*. in *INFOCOM 2002*. 2002. New York, NY USA.
9. Castro, M., et al., *Exploiting network proximity in peer-to-peer overlay networks*. 2002: Submitted for publication.
10. Zhao, B.Y., et al. *Brocade: Landmark Routing on Overlay Networks*. in *1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*. 2002. Cambridge, MA, USA.
11. Andersen, D.G., et al. *Resilient Overlay Networks*. in *18th ACM Symposium on Operating Systems Principles (SOSP)*. 2001. Banff, Canada.
12. Savage, S., et al. *Detour: a Case for Informed Internet Routing and Transport*. in *IEEE Micro*. 1999.
13. Xu, Z. and Z. Zhang, *Building Low-maintenance Expressways for P2P Systems*. 2001, Hewlett-Packard Labs: Palo Alto.
14. Roweis, S. and L. Saul, *Nonlinear dimensionality reduction by locally linear embedding*. *Science*, 2000. **290**(Theoretical Computer Science, 181): p. 2323--2326.
15. Asano, T., et al., *Space Filling Curves and Their Use in the Design of Geometric Data Structures*. 1997(Theoretical Computer Science, 181): p.3-15.
16. GT-ITM, <http://www.cc.gatech.edu/projects/gtitm/>.
17. BRITE, <http://www.cs.bu.edu/brite>.