Hybrid Overlay Structure Based on Random Walks

Ruixiong Tian, Yongqiang Xiong, Qian Zhang, Bo Li, Ben Y. Zhao, Xing Li *†

Abstract

Application-level multicast on structured overlays often suffer several drawbacks: 1) The regularity of the architecture makes it difficult to adapt to topology changes; 2) the uniformity of the protocol generally does not consider node heterogeneity. It would be ideal to combine the scalability of these overlays with the flexibility of an unstructured topology. In this paper, we propose a locality-aware hybrid overlay that combines the scalability and interface of a structured network with the connection flexibility of an unstructured network. Nodes self-organize into structured clusters based on network locality, while connections between clusters are created adaptively through random walks. Simulations show that this structure is efficient in both delay and bandwidth. The network also supports the scalable fast rendezvous interface provided by structured overlays, resulting in fast membership operations.

1 Introduction

Overlay networks are popular as infrastructures for network applications such as streaming multimedia [4], video conferencing and P2P gaming [10]. For these applications, fast membership operations and efficient data delivery are becoming basic usability requirements.

Recent developments of structured [16, 13, 20] and unstructured [18, 9] overlay networks point to a new diagram for overlay research to address these major challenges, i.e., scalability, efficiency and flexibility. Several application-layer multicast systems [14, 21] build on these structured overlays by using reverse path forwarding to construct multicast trees.

Structured overlays address the scalability requirements, but their homogeneous design can result in inefficient group communication on heterogeneous networks, by either overloading or under-utilizing resources. This impact is especially visible on bandwidth-demanding multicast services. In contrast, multicast nodes on unstructured overlays can choose the number and destinations of their connections, adapting them to network heterogeneity for improved network performance. However, unstructured overlays often require flooding or gossip to route multicast messages [9], limiting scalability and efficiency.

To combine advantages from both approaches, we propose an application infrastructure we call *Hybrid Overlay Networks (HONet).* HONet integrates the regularity of structured overlays with the flexibility of unstructured overlays in a hierarchical structure. For network locality, nodes form clusters, each providing a root node that together form a core network. Local clusters and the core network are separate structured networks. In addition, random connections between members across clusters serve as shortcuts to reduce network delay and bandwidth consumption.

We make these connections using a random walk algorithm. The number of random connections is chosen according to each node's local service capacity. The neighbors connected to by these links are chosen probabilistically according to the distribution of node service capacities through random walk. This allows these connections to adapt to network heterogeneity.

HONet is an abstract framework and can work with structured overlays such as Chord [16], Pastry [13], Tapestry [20] or De Bruijn networks [11]. In this paper, we use De Bruijn networks as an example, and describe a protocol to construct degree-constrained HONet (called HDBNet). We evaluate the performance of HDBNet through simulation and show that HDBNet is flexible and efficient. The relative delay penalty and cost in HDBNet are roughly 1/5 and 1/2 relative to a flat De Bruijn network.

The rest of paper is organized as follows. We describe the problem context and related work in Section 2. Next, we propose the HONet framework in Section 3. Then in Section 4, we present the random walk scheme to construct random connections

*This work is performed while Ruixiong Tian is a visiting student at Microsoft Research Asia. Ruixiong Tian and Prof. Xing Li are from Tsinghua University.

†Yongqiang Xiong and Qian Zhang are with Microsoft Research Asia, Beijing China, E-mail: {yqx,qianz}@microsoft.com. Bo Li is with Department of Computer Science, Hong Kong University of Science and Technology. Ben Y. Zhao is with Department of Computer Science, U. C. Santa Barbara.

between clusters. We present simulation results in Section 5, and conclude in Section 6.

2 Related work

Several approaches have been taken to address routing inefficiency and network heterogeneity in overlay networks. Techniques to exploit topology information to improve routing efficiency in flat structured overlays can be classified into three categories [3]: geographic layout as in Topologically-Aware CAN, proximity routing as in Chord and proximity neighbor selection as in Tapestry and Pastry. However, these optimizations are often limited by the homogeneous design of structured overlays.

Another approach builds auxiliary networks on top of structured overlays, such as Brocade [19] and Expressway [17]. Although these schemes are efficient for message propagation, they are not suitable for bandwidth-demanding multicast services because some nodes with high-degree will be overloaded. HONet routing similar to Brocade routing with partitioned namespaces plus the addition of randomized inter-cluster links. Canon [7], on the other hand, solves these problems by extending the flat structured overlays to a hierarchy. Canon inherits the homogeneous load and functionality offered by a flat design while providing the advantages of a hierarchy. However it can not adapt to nodes' heterogenous service capacities, a problem solved in our scheme by removing the homogeneous flat design.

Hierarchical structure is used in unstructured overlays to address efficiency and scalability issues in systems such as mOverlay [18] and Hierarchical Gossip [9]. Although they achieve good scalability, they often rely on flooding or gossip to multicast messages, resulting in less than ideal efficiency. Random walks are used in [8] to achieve good expansion of unstructured overlays.

3 The HONet framework

In this section, we describe the general HONet framework, including its structure, construction, and routing mechanisms. Note that HONet is optimized for efficient and fast membership operations, useful for quickly and efficiently joining or switching between multicast groups.

3.1 Overall structure

As shown in Figure 1, HONet is organized as a two-level hierarchy. The lower level consists of many clusters, each containing a cluster root node. The cluster roots form the upper level, the core network.

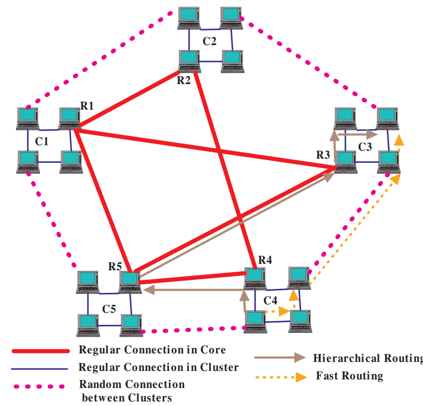

Figure 1: A HONet composed of clusters (C1,C2,...). Root nodes (R1,R2,...) from each cluster form the core network. Messages can route using the hierarchical structure of HONet, or through random connections between members in different clusters (fast routing).

The core network and each cluster are constructed as structured overlays with independent ID spaces. Each node is identified by a cluster ID (CID), which is the local cluster root's ID in the core network, and a member ID (MID), which is the node's ID in the local cluster namespace.

Node degree in HONet (*i.e.* the number of neighbors a node knows) includes two components: the regular connections in structured overlays and random connections between clusters. Each node's degree should be constrained by its service capacity and network conditions. If cluster nodes have the capacity to maintain neighbors outside of the regular overlay connections, they can create random connections with members in other clusters. We describe the construction of these random connections in section 4.

We make some general assumptions in our design: (1) The inter-cluster network latencies are larger than intra-cluster latencies, and available bandwidth between clusters is much lower than inside clusters. (2) Nodes' processing capacity and network conditions span a large range. Both assumptions are drawn from the reality of current Internet [15].

In addition to inheriting the scalable routing of structured overlays and the flexibility of unstructured networks, the clustered structure of HONet provides fault-isolation: faults in a cluster will only affect local cluster nodes, with limited impact on other clusters. Each cluster can choose the most stable member to serve as cluster root, resulting in improved global and local stability.

3.2 Node clustering

HONet is constructed through node clustering. When a node joins the network, it locates a closeby cluster to join. Node clustering provides another way to address topology-awareness in overlays.

Before a node joins, it identifies its coordinates in the network, possibly by using network coordinate systems such as GNP [12] and Vivaldi [5]. In HONet, we implement a simple coordinate system using a set of distances from each node to a group of well-known landmark nodes. The coordinates of cluster roots are stored in a distributed hash table (DHT) on the core network. A new node searches the DHT for cluster roots close by in the coordinate system.

Since most DHTs in structured overlays use a one-dimensional namespace for keys, while coordinates are multidimensional, we need to provide a mapping from the multidimensional coordinate space to the one-dimensional DHT space. The mapping should be: (1) a one-to-one mapping, (2) locality preserving, which means if two points are close in multi-dimensional space, corresponding mapped numbers are also close. Space filling curves (SFC), such as z-order or Hilbert curves [2, 17], have this property. In HONet, we use Hilbert curves to map the coordinates into numbers called locality number (L-number).

With a SFC-based mapping, to find nodes closeby to a new node coordinate l, the DHT searches the range: $X = \{x : |x - l| < T\}$. This searches for roots with L-numbers within T distance of l, where T is a cluster radius parameter. To find nearby roots, a new node sends out lookup message using l as key. The message routes to nodes responsible for l in the DHT, who search the range X, forwarding the message to closeby roots, who then reply to the new node. If the new node cannot locate nearby cluster roots, or if the distance to the closest cluster root is larger than cluster radius T, then this node joins the core network as a new cluster root and announces its L-number and its coordinates in the core network. Otherwise, the node joins the cluster led by the closest cluster root.

Since each node in a DHT maintains a continuous zone of ID space, locality information about close L-numbers (because of SFC's distance-preserving mapping) will be kept in a small set of adjacent nodes. Locating nearby clusters should be fast. Since clusters are significantly smaller than the overall network, joining a local cluster is significantly quicker than joining a flat structured overlay. Finally, a new node can add additional inter-cluster connections according to Section 4.

3.3 Message routing

If a message is destined for a local cluster node, normal structured overlay routing is used. Otherwise, we can use two approaches, *hierarchical routing* and *fast routing*, both illustrated in Figure 1. In either case, DHT routing is utilized in the core network and inside local clusters.

3.3.1 Hierarchical Routing

In hierarchical routing, messages are delivered from one cluster to another through the core network. In HONet, the MID for cluster root is fixed and a destination is identified by a (CID, MID) pair. Thus if the destination CID identifies the message as inter-cluster, the message routes to the local root first, then routes through the core network to the destination cluster's root node, and finally to the destination. This is similar to routing in Brocade [19].

Hierarchical routing is important for network construction and maintenance, and is a backup when fast routing fails. Since latencies in the core network are much larger than that inside clusters, we use *fast routing* across random connections to reduce the path delay and bandwidth consumption in the core network.

3.3.2 Fast Routing

Fast routing utilizes the random connections between clusters as inter-cluster routing shortcuts. To implement fast routing, each cluster nodes publishes information about its inter-cluster links in the local cluster DHT. For example, if a node maintains a random connection with neighbor cluster CID C, it stores this information in the local cluster DHT using C as the key. The node storing this information knows all the random connections to destination cluster C, and serves as a reflector to C.

If a source node doesn't know the random connection to the destination cluster, it simply sends the message to the local reflector through overlay routing. The reflector will decide what to do next. If it knows of nodes with random connections to the destination cluster, it forwards the message to one of them, and across the random link to the destination cluster. If the reflector knows of no such random connection, the message routes to the local root and defaults to hierarchical routing. When the message enters another cluster using hierarchical routing, it can check again to see if fast routing is available. A later reflector can tell the source node about the node with the random connection, so that later messages can avoid inefficient trangle routing. Finally, reflectors can also use its knowledge of shortcut links to balance traffic across them.

The difference between hierarchical routing and fast routing is the number of inter-cluster routing hops. Since these latencies are much larger than intra-cluster links, fast routing can significantly reduce end-to-end routing latency and bandwidth consumption between clusters.

4 Random walks

To construct random connections easily and adaptively, we use a random walk algorithm.

A node's capacity in HONet is measured by a generic fitness metric (denoted by f), which characterizes service capacity and local network conditions. According to the definition of transition probability $p_{i,j}$ in formula (1), the scale of f is not important when determining the quantity of $p_{i,j}$. Thus the fitness metric only needs to be consistent across nodes.

To consider node heterogeneity, a node's fitness metric determines the number of random connections it maintains. Our scheme samples the nodes in HONet according to the node fitness distribution. Since random walks can sample nodes according to some distribution, we propose the following algorithm to construct random connections. Assuming node i with fitness f_i has k_i neighbors, the algorithm is:

1. Node i determines the number of random connections it will create according to f_i.

2. Node i initiates a random walk message with Time-to-Live (ttl) set to s for each random connection. s is the number of skipped steps for the mixing of random walk.

3. If node i has a random walk message with $ttl > 0$, it performs the next step of random walk: select a neighbor j randomly and send the random walk message to node j with probability:

$$p_{i,j} = \frac{1}{k_i} \min\{1, \frac{f_j k_i}{f_i k_j}\} \tag{1}$$

Otherwise the message will stay at node i in next step. $ttl = ttl - 1$.

4. If node i receives a random walk message with $ttl > 0$, goto step 3. Otherwise, it is sampled by the random walk for corresponding random connection.

According to [6], we can see that above algorithm is just a typical Metropolis scheme of Markov Chain Monte Carlo (MCMC) sampling. Since the detailed balance equation

$$f_i p_{i,j} = \min\{\frac{f_i}{k_i}, \frac{f_j}{k_j}\} = f_j p_{j,i} \tag{2}$$

satisfies, if s is large enough, the obtained nodes are samples according to the distribution of fitness. Moreover, since k_i is proportional to f_i in HONet, $p_{i,j} \approx 1/k_i$, the above random walk is similar to the regular random walk in the network. For regular random walk, the mixing time is $O(\log(N)/(1-\lambda))$, where λ is the second largest eigenvalue of the transition matrix of the regular random walk [8]. Usually λ is much less than 1 and the mixing time is small for general random network if the minimum node degree in the network is large enough. Therefore the s for above random walk is small and each random connection can be created rapidly.

5 Performance Evaluation

We use a De Bruijn graph as the structured overlay network to construct HONet, and call it a Hybrid De Bruijn Network (HDBNet). The performance of HDBNet is evaluated through extensive simulations in this section.

5.1 Simulation Setup

We use GT-ITM to generate transit-stub network topologies for our simulation. We generate 5 topologies each with about 9600 nodes and 57000 edges. We assign different distances to the edges in the topologies (with values from [1]): The distance of intra-stub edges is 1; the distance of the edges between transit node and stub node is a random integer in $[5, 15]$; and the distance between transit nodes is a random integer in $[50, 150]$. Nodes in HDBNet are attached to different routers and the size of HDBNet varies from 1000 to 6000. The radix of De Bruijn graph is 2. Five runs are performed on each network topology and the average value reported.

We consider the following metrics:

- Relative Delay Penalty (RDP): the ratio of end-to-end HDBNet routing delay between a pair of nodes over that of a direct IP path. RDP represents the relative cost of routing on the overlay.

- Link cost: the average latency across all connections. The link cost is a convenient, though simplified metric to measure network structure and data delivery performance in different overlays.

- Hop count: the number of overlay hops in an end-to-end path.

5.2 Evaluation results of HDBNet

We now compare the performance of HDBNet to a flat De Bruijn overlays. The radix of flat De Bruijn

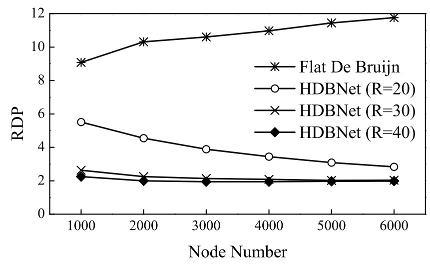

Figure 2: The average relative delay penalty (RDP) between any pair of nodes in flat De Bruijn and HDBNet when RC=4.

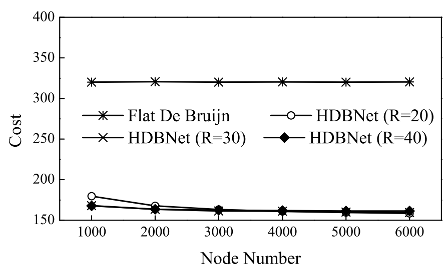

Figure 3: The link cost in flat De Bruijn and HDBNet when RC=4.

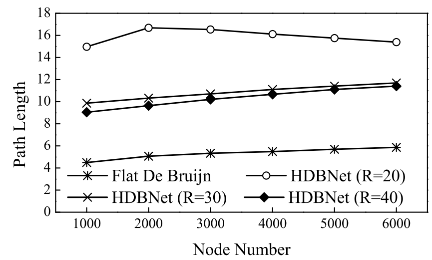

Figure 4: The average path length in term of hops in flat De Bruijn and HDBNet when RC=4.

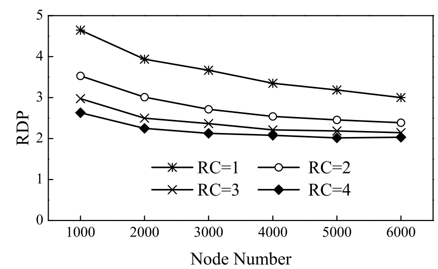

Figure 5: The average relative delay penalty (RDP) for different random connections (RC) in HDBNet when R=30.

networks is set to 4 to have similar node degrees as HDBNet. Since the cluster radius and random connections are the main factors affecting the performance of HDBNet, we first evaluate the performance with different cluster radii ($R = 20, 30, 40$) when each cluster node has at most 4 random connections ($RC = 4$). Then we compare the performance of HDBNet by varying the number of random connections ($RC = 1, 2, 3, 4$) for $R = 30$. These are representative of results run using other radius values. Fast routing is used whenever possible.

Figure 2 shows the comparison of average RDP between any pair of nodes in HDBNet and flat De Bruijn. We can see that the RDP in HDBNet is much smaller than that in flat De Bruijn which does not take the network locality into consideration. For $R = 30$ and $R = 40$, the RDP is very small (≈ 2), roughly 1/5 of the De Bruijn RDP. When the network size is fixed, RDP decreases as cluster radius grows. This is because larger cluster radii imply less clusters, and more clusters are likely to be connected directly via random links.

Figure 3 shows the comparison of link cost in HDBNet and flat De Bruijn. We can see that HDBNet has only half cost compared with flat De Bruijn, which indicates that the HDBNet is much more efficient in terms of end-to-end latency. In fact, most connections in HDBNet are intra-cluster connections, which are much shorter in term of latency than inter-cluster connections. While flat De Bruijn does not take network proximity into account, and many neighbor connections are inter-cluster links.

The comparison of average path length in term of hops between any pair of nodes in HDBNet and flat De Bruijn is shown in Figure 4. Average path length in HDBNet is 2 times or more than in the De Bruijn network. While the inter-cluster hops are reduced in HDBNet, intra-cluster hops increase. Despite this result, the path length still scales as $O(\log N)$.

Figure 5, 6, 7 show the comparison of average RDP, link cost and average path length in terms of hops respectively when $R = 30$, and we vary the maximum number of random connections per node ($RC = 1, 2, 3, 4$). The number of random connections affects performance dramatically. Just a few random connections can improve routing performance dramatically. When more random connections are allowed, messages are more likely to be delivered through random connections. Thus inter-cluster routing will be reduced, resulting in lower RDP. Moreover, fewer inter-cluster hops means less hops in intermediate clusters, resulting in shorter overlay path length. Since the intra-cluster connections are shorter than inter-cluster connections, the link cost increases with

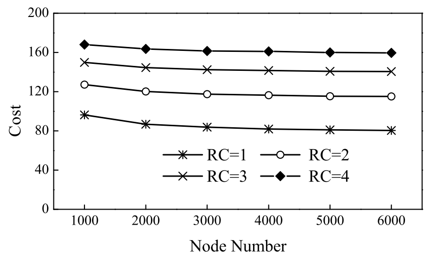

Figure 6: The link cost for different random connections (RC) in HDBNet when R=30.

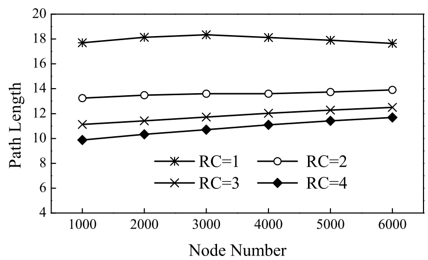

Figure 7: The average path length in term of hops for different random connections (RC) in HDBNet when R=30.

allowed random connections.

Our results show that hierarchical structured overlays can perform better than flat structures. These improvements should be applicable to HONets based on other structured overlays. Our proposed mechanisms, node clustering and random connections, offer an orthogonal way to address topology-awareness compared to locality-aware structured overlays such as Tapestry or Pastry. A performance comparison against traditional locality-aware structured overlays is part of our goals for ongoing work.

6 Conclusions

In this paper, we propose HONet, a locality-aware overlay framework for flexible application-layer multicast that combines the scalability and interface of structured networks and the flexibility of unstructured network. We use random walks to create random connections between clusters of nodes. HONet preserves the key features such as scalability, efficiency, routability and flexibility as in the structured or unstructured overlay networks, and is a desirable platform for flexible group communication.

References

[1] The pinger project. Available online at http://www-iepm.slac.stanford.edu/pinger/

[2] Asano, T., et al. Space-filling curves and their use in the design of geometric data structures. *Theoretical Computer Science 181*, 1 (1997), 3–15.

[3] Castro, M., et al. Exploiting network proximity in distributed hash tables. In *International Workshop on Peer-to-Peer Systems* (2002).

[4] Chu, Y., et al. Enabling conferencing applications on the internet using an overlay multicast architecture. In *ACM SIGCOMM* (August 2001).

[5] Dabek, F., et al. Vivaldi: a decentralized network coordinate system. In *ACM SIGCOMM* (2004).

[6] Aldous, D., and Fill J. Reversible markov chains and random walks on graphs. Draft book, available online at http://stat-www.berkeley.edu/users/aldous/RWG/book.html

[7] Ganesan, P., Gummadi, K., and Garcia-Molina, H. Canon in g major: Designing dhts with hierarchical structure. In *ICDCS* (March 2004).

[8] Gkantsidis, C., Mihail, M., and Saberi, A. Random walks in peer-to-peer networks. In *IEEE INFOCOM* (March 2004).

[9] Kermarrec, A.-M., Massoulie, L., and Ganesh, A. J. Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed systems 14*, 3 (2003), 248–258.

[10] Knutsson, B., Lu, H., Xu, W., and Hopkins, B. Peer-to-peer support for massively multiplayer games. In *IEEE INFOCOM* (March 2004).

[11] Loguinov, D., Kumar, A., Rai, V., and Ganesh, S. Graph-theoretic analysis of structured peer-to-peer systems: Routing distances and fault resilience. In *ACM SIGCOMM* (August 2003).

[12] Ng, T. S. E., and Zhang, H. Towards global network positioning. In *ACM SIGCOMM IMW* (2001).

[13] Rowstron, A., and Druschel, P. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *ACM Middleware* (Nov. 2001).

[14] Rowstron, A., Kermarrec, A.-M., Castro, M., and Druschel, P. Scribe: The design of a large-scale event notification infrastructure. In *NGC* (UCL, London, Nov. 2001).

[15] Sen, S., and Wang, J. Analyzing peer-to-peer traffic across large networks. *IEEE/ACM Trans. on Networking 12*, 2 (2004), 219–232.

[16] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM* (August 2001).

[17] Xu, Z., Mathalingam, M., and Karlsson, M. Turning heterogeneity into an advantage in overlay routing. In *IEEE INFOCOM* (June 2003).

[18] Zhang, X., et al. A construction of locality-aware overlay network: moverlay and its performance. *IEEE JSAC* (Jan. 2004).

[19] Zhao, B. Y., et al. Brocade: Landmark routing on overlay networks. In *IPTPS* (2002).

[20] Zhao, B. Y., et al. Tapestry: A resilient global-scale overlay for service deployment. *IEEE JSAC 22*, 1 (Jan. 2004), 41–53.

[21] Zhuang, S. Q., et al. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *NOSSDAV* (2001).