Virtual Direction Routing for Overlay Networks*

Bow-Nan Cheng
ECSE Department
Rensselaer Polytechnic Institute
bownan@gmail.com †

Murat Yuksel
CSE Department
University of Nevada - Reno
yuksem@cse.unr.edu

Shivkumar Kalyanaraman
IBM India Research Lab
Bangalore, India
shivkumar-k@in.ibm.com

Abstract

The enormous interest for peer-to-peer systems in recent years has prompted research into finding scalable and robust seeding and searching methods to support these overlay networks. Routing and search in these overlay networks have ranged from flooding-based unstructured techniques to structured ones mainly for popular and rate items respectively. In this paper, we propose a new method of establishing a virtual structure and introduce a technique to scalably route packets through an unstructured overlay network. We introduce Virtual Direction Routing (VDR). VDR is a lightweight and scalable overlay network routing protocol that uses the concept of virtual directions to efficiently perform node information seeding and lookup. State information is replicated at nodes along virtual orthogonal lines originating from each node and periodically updated. When a path lookup is initiated, instead of flooding the network, query packets are also forwarded along virtual orthogonal lines until an intersection with the seeded state occurs. We show that VDR achieves high reachability with relatively low seed and search packet TTL even under high network churn. We also show that VDR scales well without imposing DHT-like graph structures (e.g., trees, rings, torus, coordinate-space) and the path stretch compared to random-walk protocols is very good. The tradeoff is added latency by choosing suboptimal paths.

1. Introduction

The enormous interest for content distribution through peer-to-peer (P2P) systems in recent years has prompted research into finding scalable and robust seeding, searching and routing methods to support these overlay networks. P2P systems are attractive for several reasons including 1) its distributed nature, 2) shared overhead, 3) relatively quick response to dynamic network changes, and 4) ease of joins and leaves. One of the biggest challenges in P2P systems is information replication/dissemination and discovery in environments of high dynamism, i.e., churn. In order to locate items in a network of peers, various strategies for query propagation and information location need to be implemented. To support search queries, robust overlay networks with routing policies must be in place as P2P systems often assume an underlying overlay network.

P2P networks are broadly characterized into two major types based on whether or not strict overlay topologies are enforced: *unstructured* and *structured*. Unstructured P2P systems make little or no requirement on how overlay topologies are established and are often easy to build and maintain [5, 2], mainly due to the fact that peers do not have to inform other peers when they are joining or leaving. On the other hand, unstructured P2P systems tend to have difficulty in finding rare objects and because overlay topologies tend to move toward a power-law distribution when it comes to node degrees, load is often placed on high degree nodes. Early unstructured systems like Gnutella [6] queried for objects by simply flooding the network with search queries until an item was found. Flooding and even limited flooding techniques (e.g., normalized flooding [5] [7]), are typically prohibitive in large-scale networks as the large messaging overhead of flooding cannot be accommodated within the limited available bandwidth. Several techniques have been been examined to attempt to address the lack of scalability in flood-based techniques [2].

Because of the inherent lack of scalability in flood-based schemes, researchers have looked at several hierarchical and structured approaches [12][11]. Hierarchical approaches like Kazaa [9] relied on certain nodes to house more information and coordinate data for a specific subset. Although effective in their own right, hierarchical approaches require reorganization in the event of node fail-

*This work is supported by the National Science Foundation under grants 0627039, 0721452, 0721612 and 0230787. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

†Bow-Nan Cheng is now with MIT Lincoln Laboratory in Lexington, MA.

ure of local leader nodes. Recently, researchers have utilized novel distributed hash table (DHT) techniques to build virtual structures on overlay networks by mapping nodes to a specific structure be it a CHORD [12] or a coordinate space [11]. In these self-organizing overlay networks, neighborhood relations are more strictly controlled than in unstructured networks and search queries are propagated along the structure until a match is found. Though these DHT-based approaches can guarantee locating rare items, maintaining the structure might become impractical against high churn as repair techniques that detect a failure (i.e., a node or item leaving) and replicate the lost data or pointers incur substantial overhead [2]. Recent research has targeted to find a balance and focused on attaining "weak" but useful overlay structures (without requiring a strict/strong structure) while keeping the unstructured nature of the peers. Specific proposals used many techniques, e.g., probabilistic search [13], forcing a degree distribution on the overlay topology [7], adaptive topology establishment [2], and intelligent usage of topology and replication patterns in flood-based searches [5].

Our contribution in this paper is a scheme which is essentially unstructured but uses virtual directions to guide routing of search queries and potentially data traffic. As mentioned before, an underlying overlay network with specific routing strategies must already be in place for each of the search techniques to work. Routing issues are different from search issues in that 1) search does not deal with path selection but simply with finding objects and 2) search assumes an underlying overlay network. We present Virtual Direction Routing (VDR), a light-weight node information dissemination and routing technique in *unstructured* P2P systems. VDR places no restrictions on the underlying overlay topology and utilizes a novel concept we call *virtual directions* to provide efficient route lookup.

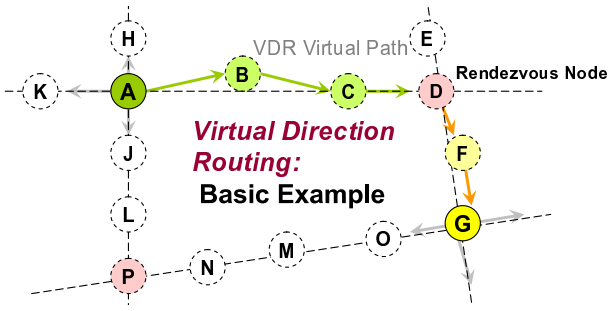

Figure 1. Virtual Direction Routing Basic Example. Node A sends data along a virtual path to rendezvous node D which then forwards it along a path along a new virtual direction to the destination Node G.

Figure 1 illustrates the basic idea behind VDR. In VDR, each node i forms a set of *virtual interfaces* ($int_i(n)$) and assigns immediate neighbors to an interface based on a hash of their unique node IDs (e.g. moded with the number of interfaces). State information is replicated at nodes along *virtual orthogonal lines* originating from each node and periodically updated. When a lookup is initiated, instead of flooding the network, query packets are forwarded along *virtual orthogonal lines* until an intersection with the seeded data occurs (the Rendezvous Node). The rendezvous node then generates a route reply packet back to source A and the data path from node A to rendezvous node D to destination G is established. If more than one neighbor is assigned to a virtual interface, ties are broken by selecting the neighbor with the ID closest to the search ID. In this way, seed and query packets automatically "gravitate" toward each other increasing the likelihood of intersect.

Key contributions of VDR include:

- Introduction of the concept of Virtual Directions to *eliminate* the need for virtual coordinate space or DHT structures to provide routing.
- A flat, scalable, and churn-resilient routing algorithm.

We will show that:

- VDR performs much better in state dissemination and reach than random walk and scales much better than flood-based techniques such as normalized flooding [5]. These two methods represent standard strategies for *unstructured* overlay *routing* (not search).
- VDR performs very well in dense networks. This is valuable as P2P overlay networks can easily form links to several other peers/nodes.
- In dynamic networks where nodes frequently go on and off, VDR significantly outperforms its counterparts in terms of end-to-end reachability and throughput.

To achieve these goals, VDR trades off the end-to-end path stretch compared to flood-based techniques. Since most real-world topologies have order $log(N)$ reach (i.e., they are less than $log(N)$ degrees from all nodes), it is expected that simple flooding techniques will find shorter paths from source to destination. VDR provides a *scalable* alternative to pure flooding and normalized flooding techniques. Additionally, state is not evenly distributed network-wide due to the biasing of dissemination packets. The rest of the paper is organized as follows: Section 2 outlines the concept of VDR including a detailed explanation of information replication and lookup. Section 3 evaluates VDR against several protocols under varying conditions of churn and TTL. Finally, section 4 presents some concluding thoughts and ideas on future work.

2. Virtual Direction Routing

The concept of Virtual Direction Routing (VDR) is simple: in flat networks, two pairs of orthogonal lines centered at different points will intersect at two points at minimum. By seeding state information along orthogonal lines and performing node lookups along those same *virtual directions*, one can ensure successful node lookup in an unstructured manner without flooding the network as these seed and search packets intersect. In this section, we outline VDR and discuss various techniques for mapping neighbors to interfaces in a globally consistent manner, requiring low maintenance manner under various topologies as well as state dissemination and lookup techniques.

2.1 Virtual Interface Assignment

In this section, we define the concept of *virtual interfaces* as used in VDR. Traditionally, interfaces are physical devices that offer points of connection between other devices. These devices can be physical connectors or wireless antennas that negotiate links between neighbors. In VDR, each node partitions its set of one hop (or low latency) neighbors into an n number of virtual interfaces. The total number of virtual interfaces per node can be fixed or varied but the partitioning strategy (i.e. hash functions) must be globally consistent. We will assume for now that the total number of virtual interfaces a node has (n) is fixed and globally consistent (i.e. all nodes decide on the same number of virtual interfaces and this number does not change).

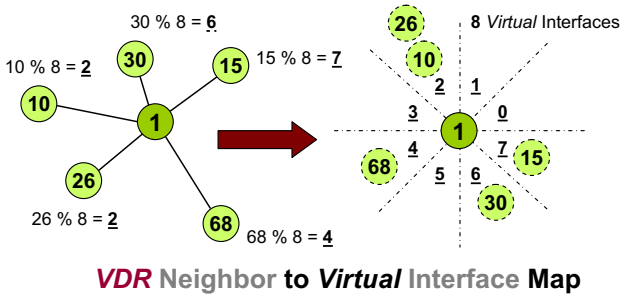

Figure 2. VDR Virtual Interface Assignment

The concept of virtual interface assignment is illustrated in Figure 2. Each virtual interface is assigned an ID from 0 to $n - 1$ and each one hop neighbor (as determined by physical neighbors or by a latency constraint) is assigned to a specific interface. In assigning nodes to an interface, it is important to keep the assignment globally consistent even in the presence of high churns. In other words, nodes assigned to a specific interface should always be assigned to the same interface even if they are unreachable for a certain amount of time. This will minimize the dynamism and make replicated data less susceptible to network changes.

Assuming each node has a unique identifier (e.g. IP address), we employ a simple heuristic to assign neighbors to an interface: 1) Hash each neighbor node ID to 160 bit IDs using SHA-1 [4] and 2) Mod the resulting value by the number of interfaces and assign the node to the interface ID with the resulting value. By assigning neighbors in the preceding manner, we are able to consistently map neighbors to the same interface despite network churn. It is important to note that with these conditions, some interfaces might have more neighbors assigned to them than others.

After all the neighbors have been assigned to a virtual interface, a *virtual north* is randomly chosen for each node by randomly selecting an interface to be the *virtual north*. This selection is important because information is later forwarded out orthogonal directions with respect to this *virtual north*. Note interface hash assignment is *NOT* the same as using DHTs to create structures. We simply use the hash for naming as our technique is *unstructured*.

2.2. State Information Dissemination

In order to minimize network flooding, each node disseminates its own ID to specific neighbors in the network to make itself easier to locate. To do this, each node periodically *seeds* its own ID to nodes along orthogonal paths with respect to its own virtual north. Each node will select 4 interfaces that are orthogonal to each other and choose the neighbor along that virtual interface which has the closest hashed ID match to the source node's hashed ID.

When the neighbor node receives this *seed* packet, it will note the previous hop and source of the packet in its routing table (storing the source as the destination and the previous hop as the next hop) and forward the packet out the interface that is *virtually opposite* of the receiving interface (until TTL is reached). The packet is not flooded to all neighbors assigned to that virtual interface, however, but the neighbor that has a hashed ID closest to the source's hashed ID. This will ensure that the packet forward is *biased* toward nodes that are closer in ID to the source so searching for nodes will form a much higher level of convergence.

A secondary heuristic (in addition to pure random walk) is used for comparison in our simulations: randomly choosing a neighbor in each virtual direction rather than biasing it toward the ID of the source.

2.3. Route Query

When a node wants to do a search for another node in the network, it generates a route request (RREQ) packet and forwards it along virtually orthogonal interfaces with

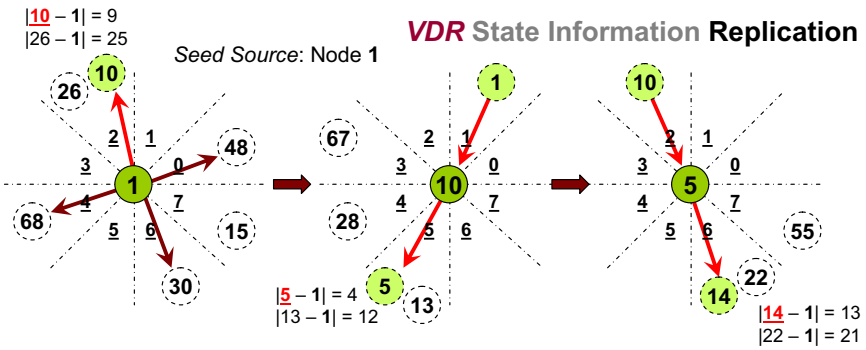

Figure 3. VDR State Information Seeding Example

respect to its virtual north. Upon receipt of the packet, each neighbor will update its routing table with a "destination - next-hop" entry based on the RREQ packet's source and previous hop and check to see if it has a routing entry to the node the source is searching for. If not, it will forward the node to the interface *virtually opposite* the receiving interface until it reaches a node that has information to the search destination or reaches its own TTL.

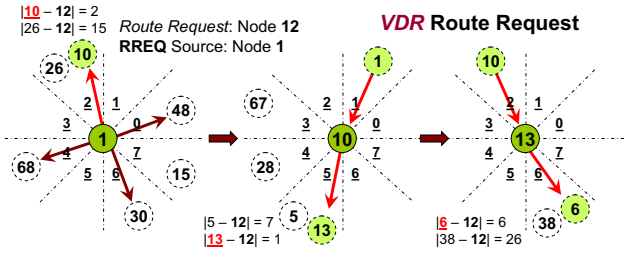

Figure 4. VDR RREQ Path Illustration: Packets are biased towards destination ID.

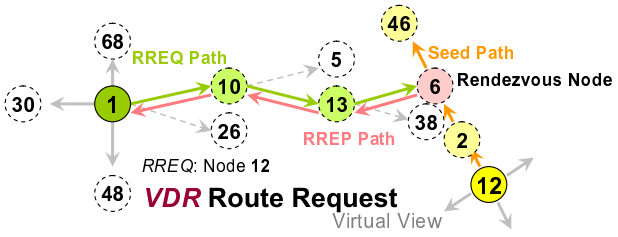

Figure 5. VDR Dissemination and Route Request Virtual View

If, however, an entry to the destination exists, the node will send a route reply (RREP) packet which contains the number of hops to the request destination in the reverse direction, relying on routing table entries of the reverse path to get back to the source. Under network churn, if a node in the reverse path is no longer active, VDR will re-select a node in the same virtual direction that has the closest hashed ID match to the original source of the RREQ packet to forward. This ensures a globally consistent biasing of the packets toward the intended destination despite path breakages due to network churn. Path deviations employ an angle correction method as described in [3].

2.4. Path Deviation

There are instances when nodes wishing to forward in a specific interface find that no neighbors are assigned that virtual interface. VDR employs a strategy to correct for path deviations in an attempt to maintain virtual straight lines. The strategy is fairly straight-forward and employs an angle correction method based on encoding a *multiplier* in the header based on the number of interfaces deviated from the intended send direction. More information can be found in [3].

3. Performance Evaluation

In this section, we provide performance evaluations of VDR under various parameters and against some basic random-walk techniques and flooding techniques. The simulations were performed using PeerSim [1] under a cycle-driven model. We wire our topology such that each node has a k out-degree. Because links are bidirectional, it is expected that each node has an average of $2k$ one hop neighbors. Although Internet topology is power-law (many nodes have few connections while some nodes have a large number of neighbors), we can assume this topology because

Table 1. Default Simulation Parameters

Parameter	Values
# Nodes / # of Virt Int.	50,000 / 8
Simulation Cycles	150
Churn percentage	0% - 50% every 5 cycles
Seed/Search TTL	10 - 100 hops
Seed Entry Expiry	10 Cycles (under churn)
Number of Queries	1000 Randomly Generated

1) peer-to-peer systems are overlay networks and connections are often virtual, 2) 1 hop neighbors can be physical one hop neighbors or links with the lowest latency, and 3) peer-to-peer systems represent a subset of the whole network and small-world examples show relatively flat topologies [8, 10].

The performance metrics evaluated include *reachability*, *path stretch vs. shortest path*, and network-wide *state distribution*. We examine these metrics under conditions of varying seed and query *TTL* and *strategies*, *average number of immediate neighbors*, *number of virtual interfaces*, and *network churn*. All simulations were averaged over 10 runs under random topologies and 95% confidence intervals were mapped. 1000 randomly generated source and destination queries starting between 30 to 100 cycles were used. Table 1 gives our default simulation parameters.

The search and seed strategies used include VDR, VDR-Random (VDR-R), and Random Walk Routing (RWR). VDR is the exact strategy described in Section 2 while VDR-Random (VDR-R) utilizes the same node to interface assignment technique, but randomizes the node forwarding in a specific direction. In short, if a virtual interface has multiple nodes assigned to it, VDR-R will choose a random neighbor associated with that interface rather than choose the neighbor with the hash closest in distance from the source node (for seed packets) or query-search node (for search packets). The random walk strategy (RWR) is not a pure random walk but each node sends 4 seed packets at periodic intervals to 4 random neighbors which "walk" the network. We call these packets "walkers". When a source node needs to find a route to a destination, 4 route request packets ("walkers") are sent out to 4 random neighbors. Each of the walkers are essentially random walk packets and are dropped after a certain TTL.

3.1. Evaluation of VDR in Churn-less Environments

In this section, we examine the effect of search and seed packet TTL, number of virtual interfaces, and average number of neighbors per node on reachability, path stretch, and state distribution under the three seed strategies as listed above (VDR, VDR-R, and RWR) in a fixed, no churn environment. Each of the 50,000 simulated nodes utilize 8 virtual interfaces with out-degree k assigned to 10 (20 neighbors).

For all cases, seed information is sent only once and the expiry time for each entry is set to the number of simulation cycles as we assume that the network is not dynamic and continual send is redundant. This is also important because under the random walk routing (RWR) technique, continual sending of the seed packets lead to different neighbors chosen each time leading to huge confusion in path choices (essentially, all nodes in the network would know a source after a set time if the expiry was set high).

3.1.1 Effect of Seed and Search TTL

In this subsection, we examine the effect of search and seed packet TTL on the metrics above. We expect that VDR should provide higher connectivity and lower path stretch than the other strategies (VDR-R and RWR) under smaller seed/query TTL simply because it biases the packets toward a specific ID.

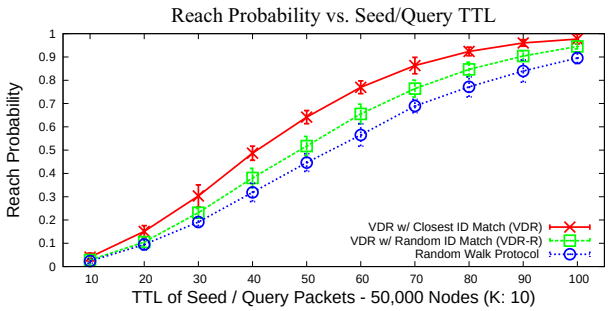

Figure 6. VDR-R achieves better reachability within less TTL in comparison to RWR. Additional consistency reinforcement with closest ID match (VDR) improves the reachability further.

It can be seen in Figure 6 that VDR is able to find nodes with a higher success rate with less query and seed TTL. This is beneficial because lower TTL lowers the amount of packets traveling network-wide, freeing up links for data. It is interesting to note that even with a TTL of 100, VDR achieves almost 100% reachability in a network of 50,000 nodes. The random walk search (RWR) technique, as expected, converged the slowest, requiring a much higher TTL to even come close to VDR. The reason that RWR even comes close to VDR is because of the fixed network environment. Under network churns, however, state maintenance would grow dramatically simply because seed dissemination would no longer be sent to the same nodes.

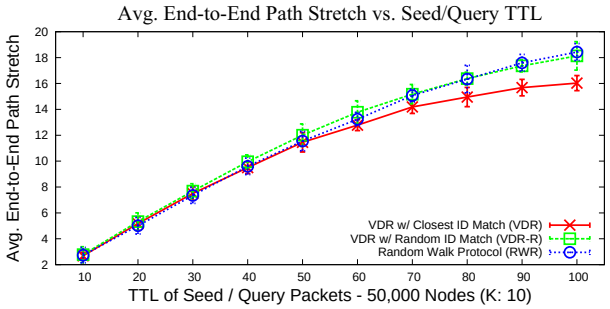

Figure 7. In VDR, path stretch from source to actual data (destination) is roughly 15% less than with random walk.

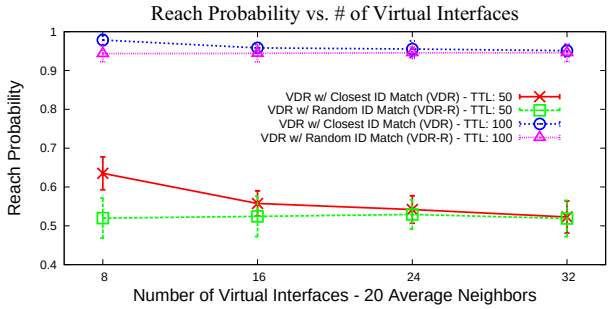

Figure 8. When the number of interfaces are less than the number of neighbors, the biasing effects of VDR are more pronounced leading to higher reach.

We see from Figure 7 that path stretch is much less in VDR. This is due to packets being biased toward the ID with the closest match. It is interesting to note the high number of hops traversed through VDR, VDR-R, and RWR as compared to shortest path. The shortest path in a wired network grows on order of $\log(N)$ where N is the number of packets in the network. Therefore, it is expected that with 50,000 nodes in the network, the shortest path should be roughly 4.7 hops. It makes sense that these path lengths increase with increased TTL because source and destination pairs that are now farther away can be reached and so the average path length increases with increased reach.

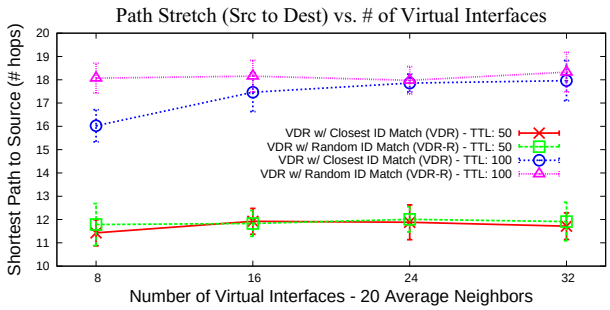

Figure 9. Biasing effects are greater seen when there are more neighbors assigned to a virtual interface, generating shorter paths.

3.1.2 Effect of Number of Virtual Interfaces

In this section, we examine the effect of modifying the number of virtual interfaces on the metrics listed earlier. With finer granularity (more virtual interfaces), it is expected that the difference between VDR and VDR-R will become smaller because the randomness in neighbor selection for each interface will be reduced as there would only be 1 neighbor per interface. Figures 8-9 show our results for simulating VDR and VDR-R with a search/seed TTL of 50 and 100.

As shown in Figure 8, VDR has much higher reach probability with lower number of virtual interfaces. This is due to the biasing of IDs such that there is a better convergence. Interestingly, when the number of interfaces increase, the probability that two neighbors will select different virtual north will be higher. This different virtual views deviate the consistency of the peers' view of the network and thus reduce the consistency of the "orthogonal" lines. The results are more pronounced at lower seed/search TTL simply because there isn't a saturation of states. The closer VDR gets to 100% reach, the less TTL will affect the packet reach probability resulting in less difference in reach. One of the reasons for greater reach is the lowered path length required for VDR as compared to VDR-R. This again, is due to the biasing of packet IDs. It is interesting that the lower the TTL, the lower the path stretch observed. This is because there is a smaller fraction of delivery success and only the paths that succeed (the shorter ones) are measured.

Figure 10 shows the *spread* of states maintained network-wide. VDR and VDR-R average around the same number of states per node. The state deviation (state spread) pictures how well distributed the states are network-wide. A smaller spread equates to a more evenly distributed network. We see that although VDR provides higher reach and better path stretches, the states are spread rather unevenly network-wide. This is a result of announcement packets constantly biasing their information to nodes with IDs closer to themselves. As a result, neighbors that haven't sent seed packets are often left with fewer states to maintain (only the ones that come in through request packets).

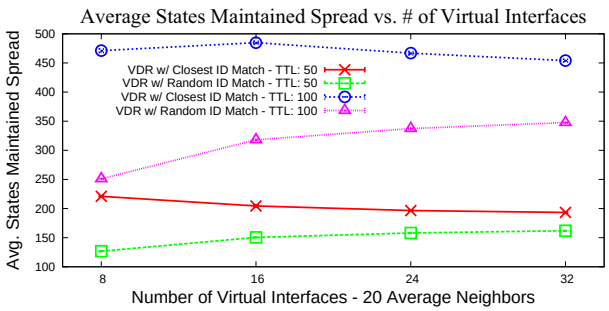

Figure 10. VDR has high state distribution deviation suggesting an uneven distribution of state networkwide.

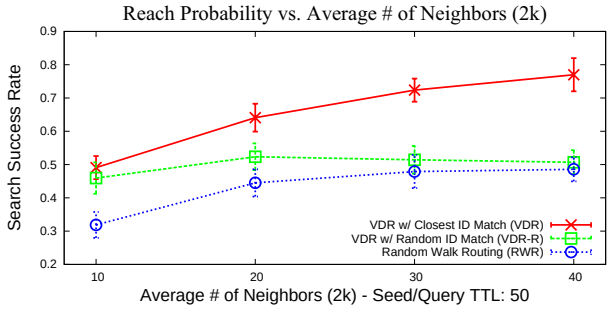

Figure 11. VDR has higher reachability than VDR-R and RWR with increased neighbors and search/seed TTL of 50 hops because of biasing packets toward the query destination.

3.1.3 Effect of Number Neighbors

In overlay networks, neighbor nodes are often assigned randomly based on the latency from a specific node rather than physical links. Because of this flexibility in neighbor assignment, it becomes interesting to examine how increasing the number of neighbors per node affects reach, path stretch, and state distribution in networks utilizing VDR, VDR-R, and RWR.

In these simulations, we fix the virtual interfaces to 8 and increase the k constant (the number of out-degrees) from 5 (avg of 10 neighbors/node) to 20 (avg of 40 neighbors/node). Because as k is increased, a greater number of neighbors will be assigned to each interface, it is expected that the biasing effect in VDR will yield much more beneficial results over VDR-R for larger k values. As the k is increased, we also expect to observe increased path stretch under lower search/seed TTL simply because the number of nodes in the network are fixed and if each node has more neighbors, paths to each node is inherently shorter (lower shortest path yielding higher path stretch). One would also expect higher reach with increased k because end-to-end paths to all nodes are essentially shorter.

Figures 11 and 12 show our results for reachability while increasing k for each of the query and seed strategies at 50 TTL and 100 TTL. It can be seen that with VDR, as the number of neighbors increase, higher reach occurs. Under the same conditions, we see that VDR-R and RWR yield significantly *less* reach than VDR. Comparing VDR to VDR-R, we see that as the number of neighbors increase, VDR-R reach remains relatively constant. This is due in part to the forwarding mechanism found in VDR-R. In VDR-R, although the number of neighbors (and thus the number of neighbors assigned to each interface) increases, its decision-making strategy is still to choose a random neighbor in a specific virtual interface direction.

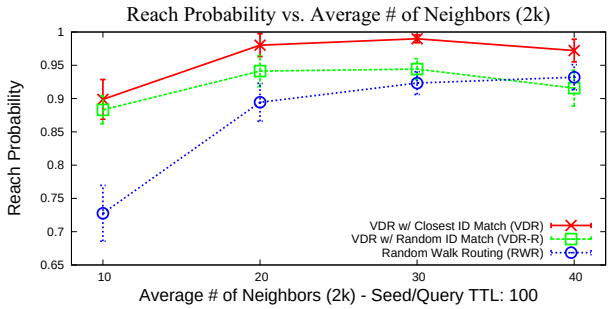

Figure 12. VDR has higher reachability than VDR-R and RWR with increased neighbors and search/seed TTL of 100 hops because of biasing packets toward the query destination.

The assignment of nodes to a virtual interface negatively impacts the options available to send and therefore the gains by simply having more neighbors (and thus shorter end-to-end paths) are offset by the losses due to assigning neighbors to rigid virtual interfaces. Because VDR-R still randomly chooses nodes in a specific interface direction, this results in a relatively constant reach even under increased k.

Figures 13 and 14 shows the results for end to end path stretch while increasing k for a query and seed TTL of 50. It's interesting that overall, the path stretch increases with increased number of neighbors. This makes sense because paths chosen are less efficient due to the greater number of neighbors assigned to each interface. Comparatively, however, VDR still yields only slightly shorter path stretch than VDR-R and RWR with increased number of neighbors.

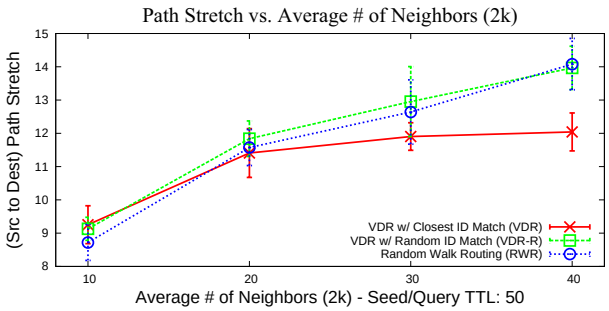

Figure 13. Path stretch increases with more neighbors because in a network of fixed number of nodes, with more connections to and from each node, the average end to end shortest path decreases.

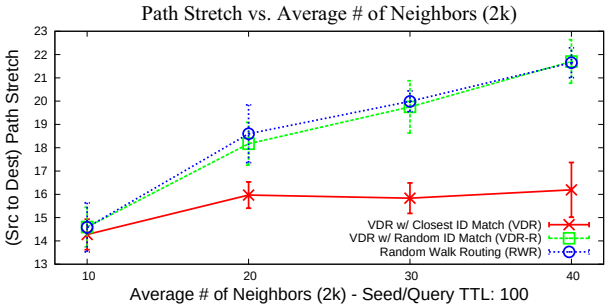

Figure 14. Path stretch increases with more neighbors because in a network of fixed number of nodes, with more connections to and from each node, the average end to end shortest path decreases.

3.1.4 Evaluation of State Distribution

Its interesting to examine how evenly the state is spread network-wide because in flat topologies, even distribution suggests no single point of failure. Because VDR is essentially a biased random-walk technique, it is expected that state is fairly evenly distributed throughout the network. To simulate state distribution, we generated a fixed overlay network with an average of 20 neighbors each. Keeping this overlay network fixed, we ran the simulation 10 times with varying initial *virtual orientations* and took snapshots of the state throughout the simulation, averaging the state per node for each run over all 10 runs. A histogram of the frequency of a average states maintained is shown in Figure 15.

As Figure 15 shows, the average states maintained is less evenly distributed in VDR compared to VDR-R and RWR.

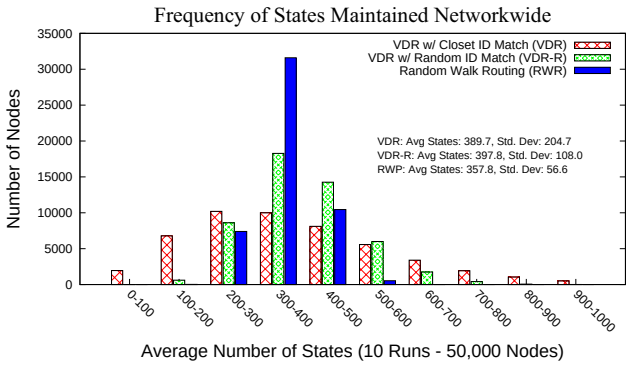

Figure 15. VDR has high state distribution deviation suggesting an uneven distribution of state networkwide.

This suggests that some nodes have more information than other nodes. We suspect this is due to certain nodes with hashed IDs closer to the average being chosen as an appropriate "next hop" more than the other nodes.

3.2. Evaluation of VDR in Dynamic Environments

In this section, we examine the effect of network churn on reachability, end-to-end path stretch, overall network load and state distribution under the three seed/search strategies as listed above (VDR, VDR-Random, and Random Walk). We simulate churn in the following manner: First, all nodes are connected by assigning an average of k out nodes from each node. Because the links are bi-directional, each node generally has roughly $2k$ neighbors. We then "turn off" half the nodes in the network probabilistically essentially dropping the average number of neighbors to k. The inactive nodes now serve as "raw material" for new connections and nodes currently in the original set can be either turned off or on per simulation cycle.

For our simulations, we fix the number of nodes active to be a constant at half the total available nodes and every 5 cycles, randomly activate a percentage of nodes with respect to the active nodes and deactivate the same number of nodes randomly. When nodes are deactivated, all the packets in their incoming queue are dropped and routing tables emptied. When they are activated, the connections that were originally formed with neighbor nodes remain the same. Thus, nodes can be active and inactive at any point in the simulation and have essentially maintain the same state.

The simulator keeps track of all the nodes that have *ever* been active and queries are generated based on any node that has *ever* been active. This makes sense as in an overlay network, resources that have never been allocated will never

be able to be found. Expiry time for each routing entry is set 10 cycles which is the same as the seed/announcement packet send interval. As per the VDR algorithm, search queries are sent out virtual orthogonal directions until they intersect a node with a path to the destination in their routing table. When this occurs, a search reply packet is generated and sent in the reverse path. In the event of reverse path nodes no longer being up, a node in the same virtual direction is chosen with an ID closest in match to the source of the search query (the destination of the search reply). Under RWR, another node is randomly chosen. In our scenarios, we simulated 25,000 active nodes with a total pool of 50,000 nodes under various churn percentages. The TTL of the seed/announcement packets was set to 150 and each node contained an average of 20 one-hop neighbors.

We examined how the percentage of network churns affect route query success. We consider a successful query to have occurred when a route request is initiated and it receives a route reply from a rendezvous node. It's expected that VDR outperforms VDR-R and RWR simply because it orders neighbors into a more structured fashion with virtual interface assignments. With the RWR, four route request packets ("walkers") are sent out to random neighbors in search for seed information planted by four seed dissemination/replication packets. These seed packets are sent out periodically to different neighbors so while at some point there might be more state network-wide, the expiry of the routing information removes stale routes quickly. Our results are shown in Figures 16 and 17.

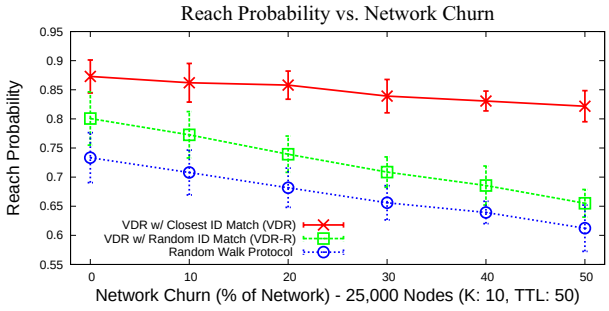

Figure 16. VDR maintains much higher reachability than VDR-R and RWR with increased percentage of network churn. It also much more robust to network churn, dropping only 5% reach for 50 seed/search TTL compared to VDR-R and RWR which dropped 12-15% going from 0% to 50% network churn for a seed/search TTL of 50.

As Figures 16 and 17 show, VDR has the highest percentage of search success/reach under the same network

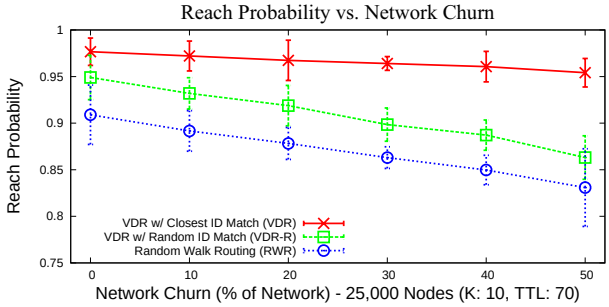

Figure 17. VDR maintains much higher reachability than VDR-R and RWR with increased percentage of network churn. It also much more robust to network churn, dropping only 2% reach for 70 seed/search TTL compared to VDR-R and RWR which dropped 7-8% going from 0% to 50% network churn for a seed/search TTL of 70.

churn rate compared to VDR-R and RWR. It outperforms VDR-R because of the biasing effect of the neighbor send. Because each node has about 15 neighbors and 8 virtual interfaces, there is a possibility that if a neighbor is down (or swapped), VDR will choose another neighbor that is atleast biased toward the search query source (search reply destination) whereas VDR-R will simply randomly choose a node. VDR outperforms RWR simply because in sending search replies, if a previous hop is no longer available, then it must randomly choose a neighbor to forward. If it was forced to perform a random walk until it reached the search query source, it would most definitely result in a packet loss the majority of the times. However, because the random walk need only intersect a node with a path in its routing table to the search query source, there is still relatively high reach (∼81% even for 50% network churn with a search/seed TTL of 70).

It is also important to understand the rate at which reach drops with respect to the percentage of churns. As can be seen from Figures 16 and 17, VDR drops only 5% in reach from 0% to 50% network churn for a search/seed TTL of 50 and only 2% for a search/seed TTL of 70. This is important because even with 50% nodes turning off and new ones being added, there is still a high degree of reach and robustness to search. VDR-R and RWR, on the other hand, drops about 12-15% in reach for 50 TTL and 7-8% in reach for 70 TTL simply because of the random nature of their send as described before: if a search query reply packet finds the next hop inactive, it must retrace its path without any kind of "hints".

3.3. Summary of VDR Performance Evaluations

Below we summarize our findings in performance evaluations for VDR:

- VDR reaches 3.5% more nodes than VDR-R and 9% more nodes than our modified random walk routing strategy (RWR).
- VDR-R produces the same reach and path stretch results with increasing number of virtual interfaces. VDR increases reach with fewer number of virtual interfaces because of its biasing technique. Gains disappear if the number of neighbors is smaller than the number of interfaces.
- Increasing the number of neighbors generally increased reach and end-to-end path stretch. This was probably due to more node choices per neighbor to bias information.
- VDR states are not well distributed.
- VDR shows a 3-4X reach retention rate going from 0% to 50% network churn compared to VDR-R and RWR, showing itself to be much more robust to network churn.
- VDR does not spread state or load evenly.
- VDR paths exhibit high path stretch compared to shortest path but good path stretch compared to pure random walk.

Most of our reported simulation results start with 8 virtual interfaces because it is the minimum required to perform angle correction. Subsequent tests with fewer interfaces (not reported in this paper) seemed to yield even better results. It seemed that when directions reflect actual physical topology (such as in wireless networks), more interfaces offer greater granularity. With virtual directions, however, fewer interfaces offer better results.

4. Conclusion

In this paper, we presented Virtual Direction Routing (VDR), a scalable overlay network routing protocol that uses the concept of *virtual directions* to efficiently perform node information dissemination and lookup. State information is disseminated to nodes along *virtual orthogonal lines* originating from each node and periodically updated. When a path lookup is initiated, instead of flooding the network, query packets are also forwarded along *virtual orthogonal lines* until an intersection with the seeded state. Both seed and search packets are biased toward the originator ID and search ID respectively. We show that in a small-world, unstructured, flat topology, VDR provides high reach even with low seed/query TTL (∼98% reach for a TTL of 100 for a 50,000 node network) and that VDR is robust to churn (dropping only 2% in reach going from 0% to 50% network churn). We also show that VDR scales well without imposing DHT-like graph structures (e.g.: trees, rings, torus, coordinate-space, etc.) and the path stretch compared to random-walk protocols (the traditional method to route in unstructured overlay networks) is very good. VDR trades off the gains by not having an even distribution of state. This is due to the biasing of state dissemination packets such that certain neighbors consistently receive state information while others do not. Path choices are also suboptimal compared to flooding techniques due to the two phased-biased random-walk nature of VDR.

References

[1] PeerSim: A Peer-to-Peer Simulator. `http://peersim.sourceforge.net`.

[2] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker. Making gnutella-like systems scalable. In *Proceedings of ACM SIGCOMM*, 2003.

[3] B. Cheng, M. Yuksel, and S. Kalyanaraman. Orthogonal rendezvous routing protocol for wireless mesh networks. *IEEE/ACM Transactions on Networking (ToN)*, 17(2):542–555, April 2009.

[4] D. Eastlake and P. Jones. Us secure hash algorithm 1 (sha1). RFC 3174, September 2001.

[5] C. Gkantsidis, M. Mihail, and A. Saberi. Hybrid search schemes for unstructured peer-to-peer networks. In *Proceedings of Conference on Computer Communications*, 2005.

[6] Gnutella Inc. http://www.gnutella.com, 2004.

[7] H. Guclu and M. Yuksel. Limited scale-free overlay topologies for unstructured peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*, 20(5):667–679, May 2009.

[8] A. Iamnitchi, M. Ripeanu, and I. Foster. Small-world file-sharing communities. In *Proceedings of Conference on Computer Communications (INFOCOM)*, 2004.

[9] KaZaA Inc. http://www.kazaa.com, 2004.

[10] J. Kleinberg. Navigation in a small world. *Nature*, page 845, 2000.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of ACM SIGCOMM*, 2001.

[12] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Tran. on Networking*, 11(1):17–32, Feb 2003.

[13] W. Terpstra, J. Kangasharju, C. Leng, and A. Buchmann. BubbleStorm: Resilient, Probabilistic, and Exhaustive Peer-to-Peer Search. In *Proceedings of ACM SIGCOMM*, 2007.