Virtual Id Routing

A scalable routing framework with support for mobility and routing efficiency

Guor-Huar Lu
luxx0137@umn.edu

Sourabh Jain
sourj@cs.umn.edu

Shanzhen Chen
schen@cs.umn.edu

Zhi-Li Zhang
zhzhang@cs.umn.edu

Department of Computer Science, University of Minnesota-Twin Cities
Minneapolis, MN-55455 *

ABSTRACT

Current flat-id based routing schemes promise support for mobility and scalability. However, providing efficient routing with minimal overheads for mobility is still a challenge. In this paper we provide a solution to these problems by introducing *Virtual Id Routing* (VIR). VIR meets these basic requirements of future id-based routing schemes: namely, i) *scalability*–by using distributed hash table based routing framework; ii) *mobility support*–by separating the node-identifier from the network location; and iii) *routing efficiency*– by exploiting network topology information, which is done by introducing a dynamic, self-organizing virtual id (*vid*) space layer in between the node id space (*uid*) and the network topology. Preliminary evaluation of the protocol shows promising results, specifically routing stretch for VIR is very close to 1, and the overheads due to mobility are also much smaller than existing schemes such as VRR.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms

Algorithms, Design, Performance, Reliability

Keywords

Network Routing, Mobility, Distributed Hash Table

1. INTRODUCTION

The popularity of Internet has made it the *de facto* information infrastructure today. However, the use of an IP address as both the identity and location of an endpoint has made current Internet architecture inadequate to incorporate new network services and functionalities such as mobility and multi-homing. Due to these shortcomings, it has been argued that for any proposed future Internet architectures [1], one should separate identity from location. In addition, such new architectures should also have the ability to persistently name objects such as endpoints, data, or services in the network, i.e., assign each object an identifier that is unique and persistent over time. It is also argued that for persistent naming, no semantics should be attached to the name (identity). Thus a *flat id space* should be used. We will be using term 'unique Id' or *uid* for our discussion in this paper which refers to a persistent identifier for every network object.

There are several advantages of using flat ids as the naming space [4]. First of all, unlike IP addresses that need to be carefully assigned to ensure uniqueness and to match the network topology, the allocation of *uid* only needs to ensure uniqueness. Second, it provides better support for mobility, as nodes do not change their *uid* when they move to a different network location. Third, use of *uid* allows easy interoperability between heterogeneous networks.

The notion of flat-id space has been widely adopted in structured peer-to-peer networks (see, e.g., [10, 6] using the DHT (distributed hash table) techniques, as well as network services built on top of such structured peer-to-peer networks (see, e.g., [9]). However, these networks and services assume the existence of IP-based network routing that provides the needed *point-to-point* connectivity between two IP addresses. While these networks and services can be used (or are built) to circumvent some of the shortcomings of the current Internet, they inherit the fundamental shortcomings that come with the existing IP addressing scheme. Inspired by the structured peer-to-peer networks, several recent proposals [3, 4, 5] attempt to directly apply DHT techniques to build scalable routing protocols on top of a unstructured flat-id space without assuming an IP-based routing infrastructure underneath it. In order to provide connectivity between two logical *uid* neighbors that may be physically far away, a number of techniques are used. For example, in VRR [3], physical paths between any pair of logical *uid* neighbors are explicitly constructed and maintained. As a consequence, these solutions may incur long routing stretches. In order to reduce the routing stretch these schemes rely on somewhat ad hoc and less scalable "workarounds", e.g VRR [3]

*The work was supported in part by the National Science Foundation grants CNS-0435444, CNS-0626812, CNS-06268 and CRI 0709048.

MobiArch'08, August 22, 2008, Seattle, Washington, USA.

uses local "short-cuts" (and two-hop neighbor information) for reducing the routing stretch. Furthermore, as nodes in VRR store complete paths to their logical neighbors, the cost of maintaining these paths can increase significantly in mobile scenarios.

In this paper we propose a novel approach in building a flat-id-based routing protocol that attempts to take advantage of the scalability of DHT-based routing while at the same time taking into the underlying network topology so as to reduce routing stretch and maintenance overheads. Instead of building a DHT-based routing protocol directly on top of a flat-id space that is "randomly" mapped onto the underlying network topology, we introduce a self-configuring virtual-id (*vid*) layer that is slotted in between the node id space and the network topology, and build our protocol on top of this virtual id space. The key idea is to have the *vid* layer which reflects the underlying network topology. Such an approach allows us to improve the protocol performance significantly compare to earlier flat-id based approaches. We adopt Kademlia [6] for routing in the *vid* space, which has better stability, minimal configuration overhead and better route selection with bounded routing distance, compared with other DHT techniques, and therefore better suited for networks with mobility.

The *vid* space embeds and reflects the underlying network topology. It is self-organizing in the sense that the *vid*'s of nodes are not persistent, and are dynamically assigned as nodes change their locations, to preserve and reflect the underlying network topology. For this reason routing protocol requires a lookup for node's *vid* using it's *uid*. Therefore we developed a mapping mechanism that maps each node's *uid* into its *vid* using the concept of hierarchical rendezvous point. This mapping mechanism is symbiotically integrated with the routing protocol as part of the data forwarding process. VIR is a novel design and can be extended to various network environments. However, the focus of this paper is to illustrate the design of VIR in a single network (intra-domain) scenario.

The rest of the paper is organized as follows. Section 2 provides an overview of our routing protocol. We present our routing protocol in Section 3. In Sec. 4 preliminary evaluation results of our protocol are presented, and in section 5 we conclude this paper.

2. BACKGROUND AND MOTIVATION

In this section we provide the motivation behind our work, as well as some background on related work.

With the advances of wireless networks such as mobile ad-hoc network (MANET) and cellular networks, the handling of mobility has become a greater concern for many researchers. To date, most solutions such as AODV [8] for MANET require a certain amount of flooding to discover topological changes. Other solutions such as Mobile IP [7] require additional addressing that cannot be easily integrated into current network architecture, in addition, the fixed home agent in Mobile IP is also a source for performance concern.

In case of Internet, one of the key pitfalls in its architecture is the use of IP address as both the identifier and the network location of an endpoint. The IP address of an endpoint changes whenever the endpoint moves in order to reflect the change in the topological position, thus making the identifier of the endpoint not persistent. For example, a TCP connection uses a quadruple that includes two IP addresses, each representing an endpoint. If the IP address of one endpoint changes, i.e., when there is mobility involved, the TCP connection fails. Clearly, to handle mobility the transport protocol should be able to refer to the endpoints independent of their network location. In addition, the host centric nature and the rigid structure imposed on IP address makes the Internet architecture inflexible in terms of adapting new services, functionalities and networks.

To address the shortcomings of the today's Internet architecture, and to gear toward a more flexible, manageable future Internet, many proposals [9, 1] consider the notion of *id-based framework* that utilizes *flat identifiers* as the application addressing space, and cleanly separates identities from network locations. These id-based frameworks work by assigning each object such as endpoints and data. in the network a unique identifier drawn from a flat id space, and build scalable and robust applications and network services using distributed hash table (DHTs [10, 6]) algorithms. For example, using DHT-based overlay network, Internet Indirection Infrastructure (i3 [9]) provides rendezvous-based communication abstraction that decouples the act of sending from the act of receiving, thus enables i3 to support a wide range of communication services.

However, most of these new functionalities for the id-based networks are limited to the application space. Though DHT is a powerful substrate in building scalable services, it still relies on the underlying native routing protocols to forward messages. In particular, when building the framework over heterogeneous networks each with its own routing protocol, the interoperability between different networks becomes a great concern. Thus, to fully embrace the id-based framework, we can extend the flat id space and use it as the addressing space for routing protocols. Flat ids ensure easier allocation and seamless integration of heterogeneous networks. In addition, by using the flat id space as the addressing space, it is possible to utilize DHT to build such an id-based routing protocol.

There have been several approaches such as VRR [3] and ROFL [4] that build id-based routing protocols using DHTs. These solutions build their routing protocols by treating the collection of unique node ids as a flat id space. We call such an id space as *node id space*. They then build the routing protocol on top of the node id space using different DHT algorithms, and then progressively build physical paths to logical neighbors in the node id space. The advantage in building the routing protocol directly on top of the node id space is that it avoids the need for a resolution service, as the id itself serves as the network location, even when node moves. Therefore when an endpoint e_a with id id_a wants to communicate with another endpoint e_b with id id_b, all e_a needs to do is to send packets directly addressed to id_b without any need to discover e_b's location. However, since node id space has no inherent structure, and bears no resemblance to the underlying network topology, it is possible that the protocol's performance will suffer from long routing stretch. Such a performance degradation is due to the way DHT operates. In DHT when a node s sends a message to an id id_k, the message is routed via several (logical) hops before reaching the destination, a node d with an id id_d closest to id_k[1]. At each hop, the message is forwarded to an intermediate

[1]The closeness is defined by each individual DHT algorithm.

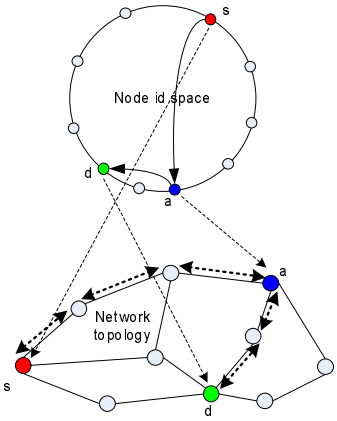


Figure 1: An example of long stretch caused by uncorrelated node id space and the network topology.

node that has id closer to id_k than the current node's id. In an overlay network, the underlying routing protocol provides point-to-point connectivity between nodes. Therefore, the physical location of nodes does not matter. However, if the physical node locations have no relation to their ids, as in the case of node id space, it is possible that when s sends a message to d via a node a, a maybe far away in physical hop distance to d than s even though a is closer to d in the node id space, and therefore it takes a longer path compare to available shortest path. Fig. 1 illustrate this scenario.

In this work we want to design a new routing framework for id-based routing. In particular, we want to avoid the long routing stretch caused by unstructured node id space. Clearly it is not sufficient to rely on the node id space to build the routing protocol as it does not take advantage of the underlying topology information. However, if we build the protocol based on the addresses derived from the topology, e.g., IP addresses, we again has a rigid address space that is not particularly suited to handle mobility. Thus, the key idea in our solution is to introduce a *virtual id space* that lies in between the node id space and the network topology, i.e., we assign each node a unique virtual id in addition to its unique id. We illustrate this concept in Fig. 2.

Our goal is to embed topological information onto this virtual id space, while at the same time inherit advantages of the id-based routing. In Fig. 1 an example of long stretch caused by uncorrelated node id space and the network topology is shown. Using shortest path routing the hop distance between s and d is 2, but using id-based routing the hop distance becomes 5 as packets need to traverse to a first. In this case the stretch is 2.5. It is clear that one of the reasons for having long routing stretch is that there is no correlation between the node id space and the network topology. Thus, in our design we want the virtual id space to have the property that if two nodes' virtual ids are close in logical distance, then they should also be physically close to avoid the potentially long routing stretch. In addition, by using a virtual id space layer, we can allow unified addressing across heterogeneous networks. Also, the construction of the virtual id space can be flexible to match different types of networks.

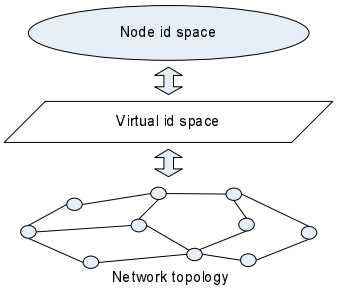


Figure 2: Virtual ID layer which maps the network topology to Node id space.

3. VIRTUAL ID ROUTING

In this section we present an overview of our virtual routing protocol. As mentioned in Sec 2, our goal is to have an intermediate virtual id space layer located in between node id space and the network topology in such a way that the virtual id space reflects the topology. The protocol contains three components: i) the construction of the virtual id space layer, ii) the routing protocol using virtual ids as addressing space, and iii) a mapping mechanism that maps unique ids into virtual ids.

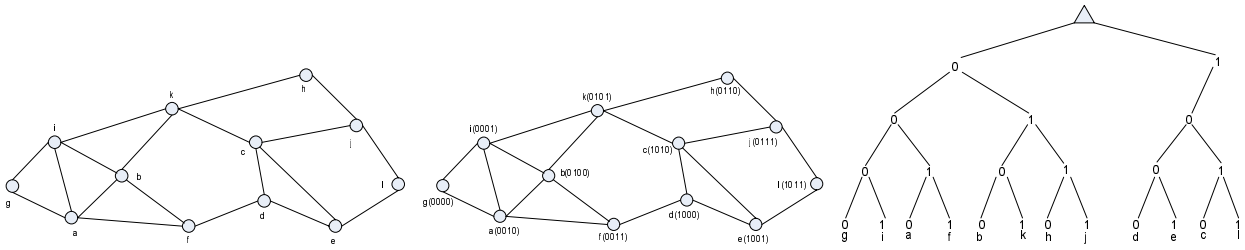
We assume each node i in the network has a unique id $u(i)$ and a virtual id $v(i)$. For the sake of simplicity we sometimes write i in place of $u(i)$. We define $d(i,j)$ as the physical hop distance between i and j. We define $p(i,j)$ as the length of the longest common prefix between i and j's virtual ids. For an l-bit virtual id space $\mathcal{S}_l$, the *logical distance* between i and j is defined as $\delta(i,j) = l - p(i,j)$. For example, if $l = 4$ and i and j's virtual ids are 0110 and 0111 respectively, then $\delta(i,j) = 1$ (as $p(i,j) = 3$).

We describe the details of each component in the rest of this section.

3.1 Virtual Id Space Construction

Existing schemes such as VRR [3] treat the collection of all unique ids in the network as a *node id space*, and build their routing protocol on top of this node id space. Since node placement is random, the node id space bears no relationship with the network topology. As a consequence, the routing stretch can be increased unnecessarily. The key idea in our design is to insert a virtual id layer $\mathcal{S}$ between the node id space and the underlying topology such that $\mathcal{S}$ closely resembles the network topology. That is, if the logical distance between two nodes i and j is small, the hop distance between i and j is also small, and vice versa. More precisely, assuming we have a network $\mathcal{G} = (V, E)$, with V as the set of nodes in the network and E as the set of all edges in the network, our goal is to have a mapping $\phi : V \rightarrow \mathcal{S}$ such that $d(i,j) \leq c \cdot \delta(i,j)$, $c \geq 1$.

Our idea in building the virtual id space is based on *clustering* techniques that are commonly used in mobile ad-hoc networks. There are distributed and scalable algorithms suitable for our protocol, such as [2]. The basic idea in building the id space is to recursively cluster a number of nodes at a time, and assign certain bits of virtual ids to the clustered nodes. For example, assuming initially all nodes in the network are level-0 nodes. If two nodes are close to each other then these two nodes will form a level-1 node, and we can assign 1-bit virtual ids (0 and 1) to these two



(a) Before virtual id assignment (b) After virtual id assignment (c) Binary-tree representation of the virtual id space

Figure 3: A simple network topology with twelve nodes.

nodes Next, we cluster two level-1 nodes to form a level-2 node, and assign additional 1-bit virtual ids to all the nodes within the level-2 node by prepending the newly assigned ones to the existing ones, and so on and so forth. The process repeats until every node in the network is assigned a unique virtual id. Similarly, we can cluster m nodes at once and assign k-bits to these m nodes such that $2^k \geq m$. Fig. 3 shows the results of the virtual id assignment for a twelve node network. We evaluated our clustering based virtual id assignment schemes on multiple topologies with various values of m. Fig. 4 summarizes one of the test results for a topology using 400 nodes for $m = 4$.

The intuition behind such an approach is simple: every time we cluster nodes or multi-nodes, we make sure the hop distance between clustered nodes are bounded. By assigning fixed bits of virtual ids to clustered nodes, we can make sure that the "closeness" property is satisfied. In addition, by building the virtual id space from the bottom-up, two nodes can start communicating as soon as they are in a same multi-node. This is particularly suitable in a heterogeneous environment where each network can build its own virtual id space and start communicating within the network. When different networks need to communicate with each other, all we need is to prepend more bits to virtual ids so that protocol can address nodes on different networks.

3.2 Routing Protocol

The virtual id space construction gives us an l-bit virtual id space $\mathcal{S}_l$ that has a m-ary tree structure (cf. Fig. 3(c)). Here we explain our routing scheme using a binary tree structure. This allows us to deploy Kademlia-like algorithm [6] for the routing protocol. However, DHT only provides us part of the solution as there are still challenges we need to overcome in order to build a DHT-based routing protocol. Following Kademlia's description, each node i in the network needs to have at least one logical neighbor j with $\delta(i, j) = 1 \ldots l$, as long as such a j exists in the network. We refer to this condition as the "*logical invariant*". As an example, refer to Figure 3, for node f with $v_f = 0011$ to satisfy the logical invariant, f must have logical neighbors with prefixes $\{0010\}$ ($dist = 1$, for which node a qualifies), $\{000\}$ ($dist = 2$, both g and i are suitable), $\{01\}$ ($dist = 3$, nodes $\{b, h, k, j\}$ are suitable), and $\{1\}$ ($dist = 4$, nodes $\{d, e, c, l\}$ are suitable). Note that this relationship is symmetrical, that is if a node i is another node j's distance-k neighbor, j is also i's distance-k neighbor.

If we have point-to-point connectivity between nodes, the logical invariant is sufficient for any two nodes to communicate. For three nodes a, b, c, if $\delta(a, b) = k$ and $\delta(a, c) = k$, then $\delta(b, c) \leq k - 1$. This is because b and c both have the same logical distance k to a, that means b and c both share the same $l - k$ bits of the prefix as a, but both had opposite bit to a immediately after the first $l - k$ bits, therefore $\delta(b, c) \leq k - 1$. Based on this observation, for a node i to lookup a node v that is some distance-k away in an overlay network, it would take k steps for i to find v if every node in the network satisfies the logical invariant. Thus, we define $i \overset{k}{\rightsquigarrow} v$ as node v is *reachable* from i in at most k logical steps.

Our routing protocol works as follows. Each node i maintains entries to its logical neighbors in its routing table rt_i. Assuming the routing table of each node in the network satisfies the logical invariant, the forwarding algorithm becomes similar to Kademlia's lookup algorithm. When a node s wants to send a packet to some destination d, assuming s knows d's virtual id $v(d)$, s inspects rt_s and finds a logical neighbor n such that $\delta(s, n) = \delta(s, d)$, since $\delta(n, d) < \delta(s, d)$. s then forward the packet to n following some pre-established routing path P_{sn}. When n receives the packet from s, n again inspects rt_n and selects a logical neighbor n' such that $\delta(n, n') = \delta(n, d)$, and forwards the packet to n'. The process repeats until no further progress can be made.

For forwarding to work, it is clear that we need some way to establish a physical path between a node i and its logical neighbor j. To address this issue, we construct the routing table in a round-by-round manner. Starting from round-1, at each round-k, a node i searches for a (logical) distance-k neighbor to be added to rt_i. The logic behind this construction is because of the the way our virtual id space is constructed, nodes that are close in terms of the logical distance are also physically close. Thus, when a node i searches for a (logical) distance-1 neighbor j_1, node j_1 is usually a 1-hop neighbor of i, or at most a few hops away. Once i adds j_1 as its distance-1 neighbor, i can begin searching for a distance-2 neighbor j_1 among itself and j_1. If j_2 is a 1-hop neighbor of i, i can add j_2 to rt_i immediately. Otherwise, if j_2 is a 1-hop neighbor of j_1, i can add j_2 to rt_i with the next hop being j_1. By repeating this process in a round-by-round manner, each node can build routing paths to its logical neighbors progressively.

3.3 Mapping and Forwarding

Although building the routing protocol on top of the structured virtual id space allows us to overcome limitations of previous approaches, it has one drawback: since virtual ids are not persistent, we need a mapping mechanism that maps

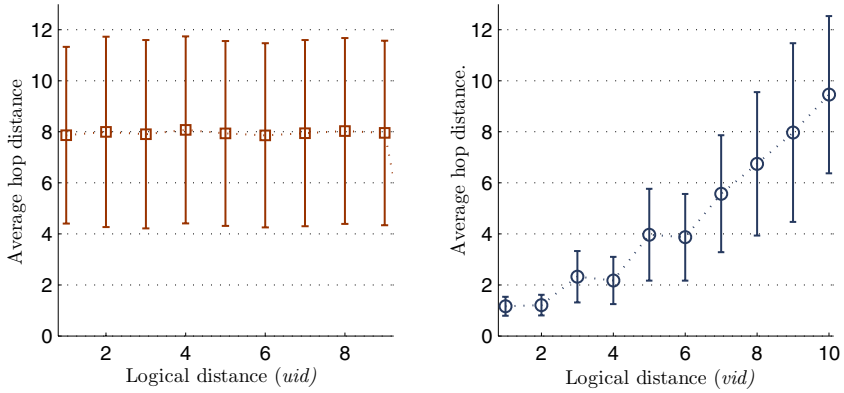


Figure 4: Comparison of hop-distance distribution for the nodes using *uid* and *vid*. (left) Distribution of average physical hop distance with respect to logical distance in *uid* space,(right) Distribution of average physical hop distance with respect to logical distance in *vid* space.

each node's unique id into its virtual id. However, unlike DNS which is a separate infrastructure, we design our mapping mechanism as part of the forwarding process. For efficiency and scalability, we design the mapping mechanism based on the idea of hierarchical rendezvous points.

The idea of hierarchical rendezvous points is as follows. For a node i with a mapping $M_i = \langle i, v(i) \rangle$, i selects several rendezvous points in the network at a progressively larger area with respect to i itself and stores M_i at these rendezvous points. We call these rendezvous points as *mapping points* (*mp*s). The area here is defined in terms of the virtual id space. The motivation in selecting a *mp* at each size of the area is that if a node is closer to i, its query message does not need to travel far in order to obtain i's virtual id, thus enabling locality-aware mapping mechanism.

Since node failures are not uncommon in wireless scenarios, in our design, for each node, a number of hierarchical rendezvous points are selected to store its mapping information to avoid single-point-of-failure at mapping look-up. Our virtual id space has a binary tree structure, it allows us to easily determine where to select the mapping points. For l-bit virtual ids, we say the entire id space is a level-l space consists of all nodes in the network. For i with $v(i) = a_l a_{l-1} \ldots a_2 a_1$, the level-$(l-1)$ space (with respect to i) consists of all nodes starting with prefix $\{a_l\}$, and level-$(l-2)$ space consists of all nodes with prefix $\{a_l a_{l-1}\}$, and so on and so forth with a level-1 space consists of nodes with prefix $\{a_l a_{l-1} \ldots a_2\}$. For example refer to Fig. 3(c), node g can select one mapping point in a consistent way in each level of the virtual id space: {000} (level-1), {00} (level-2), {0} (level-3), and all nodes with prefixes of either {0} or {1} (level-4).

To look up i's mapping, a node j simply computes i's potential mapping point (*pmp*) with respect to its space starting from level-1. If i's information cannot be find, j computes a level-2 *pmp* for i and so on and so forth, until i's information found at level-k *pmp*, which means that both i and j reside in the same level-k space. It is clear that as long as both i and j reside in the same level-k space, a query packet from j never needs to traverse beyond i's level-k *mp*, i.e., the query cost is bounded by smallest level of the virtual id space both i and j resides in.

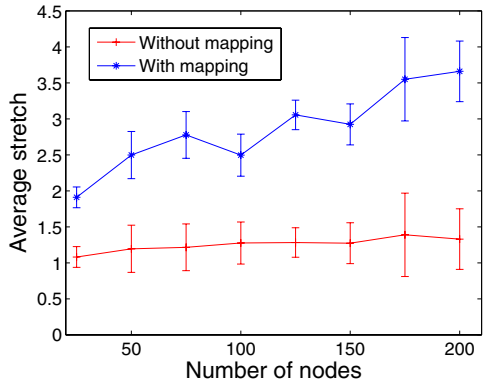


Figure 5: Average stretch of our routing protocol.

The combined mapping and forwarding works as follows. When a node j needs to communicate with i, it first needs to obtain i's virtual id so packets can be delivered to i. To do so, the first packet from j is a special query packet containing j's virtual id $v(j)$. The query packet is first sent to i's potential mapping points computed by j. Once the query packet arrives at a node with i's virtual id, it is forwarded to i. When i receives the query packet from j, i sends a reply packet containing its own virtual id $v(i)$ back to j using $v(j)$. When j receives the reply packet, i and j can communicate with each other directly using their virtual ids without going through additional mapping process.

4. PRELIMINARY EVALUATION

In this section we present some preliminary results for our routing protocol.

We evaluate the routing stretch of our routing protocol under different network settings. The routing stretch is defined as the ratio of number of physical hops taken by the protocol to reach from source to destination node and the shortest distance between the source and destination in terms of physical hops. We choose the routing stretch as the metric for comparison because this is one of the key concerns we

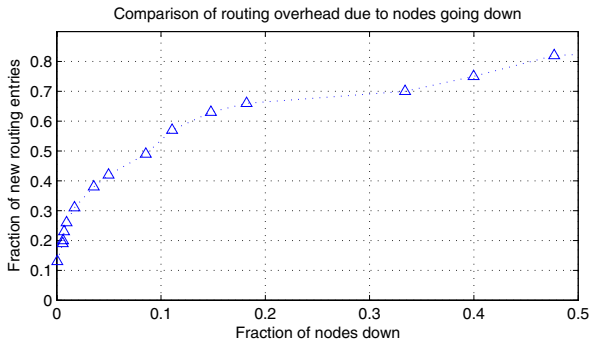


Figure 6: Routing table construction overhead for the wireless scenario.

had when we introduced the intermediate virtual id space. In addition, we wanted to compare the routing stretch of our protocol with VRR as it represents the state-of-the-art in the development of id-based routing protocol, unfortunately code for VRR is not publicly available, and currently we are in the process of implementing VRR based on the description provided in the paper. Therefore results are not provided in this paper for the comparison.

We evaluate our routing protocol under two settings: 1) first, we assume prior to sending the packets, each source knows the virtual id of the destination. This allows us to evaluate the protocol performance based on virtual ids only; and 2) we assume each source only knows the unique id of the destination and has to go through the mapping process in order to deliver packets to the destination. This evaluates the impact of the mapping process on the routing stretch. We vary the network size from 25 nodes to 200 nodes, and then randomly select different source s and destination d pairs to measure the routing stretch. Fig. 5 shows the average routing stretch with and without mapping for various sizes of the network. As we can see, when there is no mapping involved, the routing stretch increases slowly and remains pretty close to 1 (1.4) for a 200 node network.

When there is mapping involved, the routing stretch increases drastically as the network size increases. This is not surprising as the mapping mechanism requires a node to recursively query various mapping points before reaching the destination. However, since the mapping is only used before two nodes initiate communication, the impact is reduced for long-lived connection. In addition, various caching techniques can further reduce the routing stretch for the mapping.

Wireless Scenario: In the wireless environment nodes can join and leave more often. In these scenarios routing protocol will have to compute the new routing tables for the nodes every time a node goes down. In order to study the stability of the virtual id scheme, we setup a simulation environment. In the simulation, random nodes are disconnected from the network, and the percentage of total entries changed in the routing tables is calculated. Using simulation results we show that the overhead for the recomputation of routing table entries is very small for our routing protocol. Fig. 6 shows the routing overhead for our protocol in wireless scenario.

5. CONCLUSION

Id-based routing is an important step toward the routing framework for future networks. Although current id-based routing schemes provide excellent support for scalability and mobility, still they suffer from poor routing efficiency and higher overheads for the mobility. Furthermore they rely on ad hoc solutions to enhance the routing efficiency which complicates the routing framework. In this work we present a novel approach in building an id-based routing to enhance the routing efficiency of such protocols. This is achieved by allowing a self-configuring virtual id layer to embed the network topology. Preliminary evaluation demonstrates promising results for better routing efficiency with much lesser overheads for mobility. We are currently conducting more simulations and experimenting with real implementation to provide an extensive evaluation of the proposed routing framework.

6. REFERENCES

[1] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A layered naming architecture for the internet. In *SIGCOMM '04*, New York, NY, USA, 2004. ACM Press.

[2] S. Basagni. Distributed clustering for ad hoc networks. *Parallel Architectures, Algorithms, and Networks, 1999.(I-SPAN'99) Proceedings. Fourth InternationalSymposium on*, pages 310–315, 1999.

[3] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron. Virtual ring routing: network routing inspired by dhts. In *SIGCOMM '06*, pages 351–362, New York, NY, USA, 2006. ACM Press.

[4] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica. ROFL: routing on flat labels. In *SIGCOMM '06*, pages 363–374, New York, NY, USA, 2006. ACM Press.

[5] B. Ford. Unmanaged internet protocol: taming the edge network management crisis. *SIGCOMM Comput. Commun. Rev.*, 34(1):93–98, 2004.

[6] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *Proceedings of IPTPS02*, March 2002.

[7] C. Perkins. Ip mobility support for ipv4, rfc 3344. 2002.

[8] C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[9] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. *IEEE/ACM Trans. Netw.*, 12(2):205–218, 2004.

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