Tunnel Vector: A New Routing Algorithm with Scalability

Cheng-Jia Lai *
FX Palo Alto Laboratory, Inc.
Palo Alto, California, United States
cjlai@fxpal.com

Richard R. Muntz
Computer Science Department
University of California, Los Angeles
Los Angeles, California, United States
muntz@cs.ucla.edu

***A*bstract—Routing algorithms such as Distance Vector and Link States have the routing table size as Ω(n), where n is the number of destination identifiers, thus providing only limited scalability for large networks when n is high. As the distributed hash table (DHT) techniques are extraordinarily scalable with n, our work aims at adapting a DHT approach to the design of a network-layer routing algorithm so that the average routing table size can be significantly reduced to O(log n) without losing much routing efficiency. Nonetheless, this scheme requires a major breakthrough to address some fundamental challenges. Specifically, unlike a DHT, a network-layer routing algorithm must (1) exchange its control messages without an underlying network, (2) handle link insertion/deletion and link-cost updates, and (3) provide routing efficiency. Thus, we are motivated to propose a new network-layer routing algorithm, Tunnel Vector (TV), using DHT-like multilevel routing without an underlying network. TV exchanges its control messages only via physical links and is self-configurable in response to linkage updates. In TV, the routing path of a packet is near optimal while the routing table size is O(log n) per node, with high probability. Thus, TV is suitable for routing in a very large network.**

***Keywords*—routing, tunnel vector, scalability, DHT**

I. Introduction

The Distance Vector [6] and Link State [10] routing algorithms are primarily used in the current Internet infrastructure to route IP packets towards destination addresses, with a routing table size of $\Omega(n)$ per node where n is the total number of destination identifiers for routing. With hierarchical routing, Internet routers actually route packets by IP address prefixes, with a loss of mobility, i.e. a trade-off favoring scalability since the number of address prefixes is much less than the total number of IP addresses. However, these routing algorithms are not scalable for very large networks that will occur in the future, e.g., IPv6 [2] where the addresses have 128 bits and the number of address prefixes to be routed in the backbone is huge. Thus, adapting the emergent distributed hash table (DHT) techniques [4][7][8][9][13][14][16][17][19] to the design of a large-scale network-layer routing algorithm suggests opportunity to achieve significant scalability and mobility in the network backbone. Although the original design goal of a DHT was to forward queries to objects based on a specified *object identifier* (OID) that is independent of location, the overlay network maintained by a DHT is capable of routing messages to nodes by a specified *node identifier* (NID) that is independent of node location. The average size of a routing table at a node is typically $O(\log n)$, where n is the number of NIDs. Thus, adapting the technique of building the DHT overlay network to the design of a network-layer routing algorithm may achieve similar scalability and mobility.

Unfortunately, the fulfillment of such a hope requires addressing certain problems. We have identified three major challenges as follows.

1. Messaging without an underlying network
2. Handling linkage updates
3. High routing efficiency

First, in the absence of an underlying network, it is difficult for a DHT to maintain its overlay network in a predefined link structure such as a hypercube, rings, or a tree that is required. To maintain the structure of overlay links in a DHT, nodes must exchange some control messages, and the DHT exploits the existence of an underlying network that delivers its control messages to any node identified by a specified node address. However, this advantage is lost when a DHT method is adapted to a network-layer routing protocol that itself is designed to function as that underlying network, on top of data-link-layer protocols such as Ethernet.

Second, most DHT approaches handle *node* insertion and deletion, while a network-layer routing algorithm must handle insertion and deletion of physical links between nodes. The latter is more generalized since adding a node must include adding one or more links, but adding a link may only create a loop, typically for higher connectivity and network throughput. Thus, without support for handling such linkage changes, a DHT approach cannot be adapted to the design of a network-layer routing algorithm.

* This work was previously completed in Computer Science Department at University of California, Los Angeles.

Third, the latency in forwarding a datagram packet to the destination node is typically a significant factor of performance in the network layer. Thus, if a DHT method such as CAN [14] and Chord [19] only provides routing efficiency in terms of *overlay hops* instead of latency, applying the method to network-layer routing can result in poor performance and low throughput. On the other hand, a DHT method such as Tapestry [8] and Pastry [17] that optimizes the link structure by taking into account network distances in order to provide a short routing latency may suffer from the variability of network distances at run time due to traffic and queue lengths [11] if it fails to adapt its link structure accordingly.

We believe that the three challenges described above are the primary difficulties to be overcome to design a network-layer routing algorithm based on a DHT method. The first challenge (messaging without an underlying network) is the kernel of the network-layer routing problem and is actually the most difficult. PeerNet [3] has been proposed to address this problem based on the assumption that a node can be assigned a hierarchical *address* dynamically by its geographic location while the destination of a packet is specified by a *node identifier* unrelated to the node address. Then, it builds an underlying network using the node address hierarchy. We feel this scheme is impractical since if node addresses could be assigned hierarchically, the routing problem would have been solved.

The remainder of this paper is organized as follows. Section II describes the principle of our new routing algorithm. Section III defines the notation of a routing problem. Section IV describes our algorithm in detail. Conclusions and future work are listed in section V.

II. Principle

In general, to adapt a DHT method to a network-layer routing algorithm, a tunneling mechanism can be employed that implements virtual circuits (VC) for overlay links. We discuss two approaches. The first approach is to store information at all intermediate nodes on the path of a VC so that the intermediate nodes can properly forward a packet passing along the VC. However, a DHT on a network with n nodes and a diameter d typically creates $\Omega(n)$ overlay links each of which passes $\Omega(d)$ nodes. Thus, the average number of VCs passing a node is $\Omega(d)$, resulting in an average routing table size of $\Omega(d)$, which is not scalable with $d = \Omega(n^{1/2})$ as the network backbone topology is typically a mesh. The other approach to tunneling is to employ *source routing* that embeds the VC path information in each packet header so that an intermediate node simply retrieves the embedded path from the header of an incoming packet to forward it. However, this can largely increase the size of a packet header. If a source-routing path contains $\Omega(d)$ nodes, the header size of a packet passing along the VC is $\Omega(d)$, which is very inefficient.

Thus, we are motivated to propose a new network-layer routing algorithm, Tunnel Vector (TV), which maintains a conceptual overlay network in hierarchy and implements overlay links by source routing with a special design. The hierarchy has $O(\log n)$ levels. At each level k, nodes are grouped by NID prefixes of $(k-1) \times B$ bits where B is a small integer. Then, in each group, the nodes create overlay links between neighbors. Tunneling information for these overlay links is exchanged within a group by a mechanism similar to the Path Vector algorithm [18] that is now used in BGP-4 [15] for inter-domain routing on the Internet backbone. However, in TV, the destination identifier within each group has only 2^B possibilities. With random node addresses, a source-routing path would contain only $O(2^B)$ node addresses with high probability, resulting in an average routing table size of $O(2^B \log n)$ with high probability. Moreover, since the overlay links are built between neighbors within each group, the latency in forwarding a packet to the destination is near optimal.

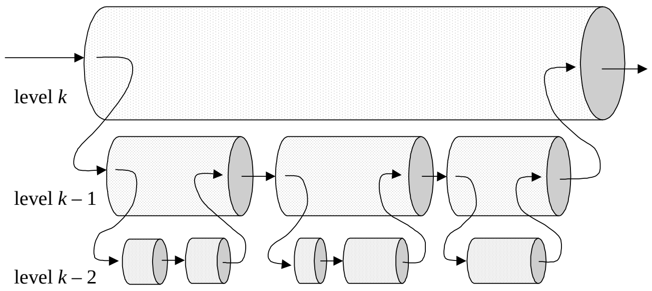

Figure 1 – Recursively tunneling until level 1

In addition, referring to Figure 1, messaging in TV at a level k is achieved by tunneling via overlay links at level $k-1$. Note that an overlay link at level $k > 1$ is implemented by a tunnel that is composed of overlay links at level $k-1$, while an overlay link at level 1 is composed of only physical links. Thus, a message at any level k can be forwarded to the destination node via only physical links by recursively tunneling until level 1 in the absence of an underlying network. Note that a packet routed by TV can simultaneously pass multiple tunnels each of which is at a different level. Thus, a "source-routing stack" is added to the packet header to carry source-routing paths for these tunnels. Since a source-routing stack is of size $O(2^B \log n)$ for $O(\log n)$ levels with high probability, the increased size of a packet header is limited, and the overhead of the forwarding mechanism is small.

III. Definition

Consider an undirected graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. A vertex is a *node* and an edge is a *link* in the target network that requires a routing algorithm. A node $v \in V$ is identified by a unique address denoted by $A(v)$. Each address has $B \times L$ bits where B and L are positive integers. A link $e \in E$ is bidirectional between two nodes u and v, with a cost $C(u, v)$ from u to v and a cost $C(v, u)$ from v to u, where $C(u, v)$ is positive if $u \neq v$, or zero if $u = v$.

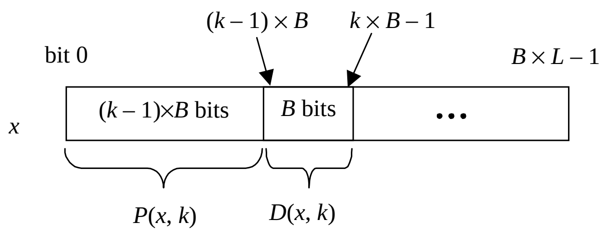

Figure 2 – Digits and prefixes of an address *x*

The routing problem is defined as follows. Given a packet q originating at a node s and with a destination address $A(q)$, q should be forwarded to a node d such that $A(d) = A(q)$ if d exists. Otherwise, q is discarded. The total cost of forwarding q from s to d is the sum of the costs of all those links traversed on the path. A shortest path from s to d is defined as a path with the minimal cost from s to d, which may be not unique. It is then desirable to forward packet q from s to d on a *relatively short path*, i.e. a path that has a total cost equal or comparable to that of the shortest path.

Refer to Figure 2. For an address x, we define $D(x, k)$ as the "routing goal" for x at a level k ($1 \le k \le L$), where the numeric value of $D(x, k)$ is the binary number represented by the B bits from bit $(k-1) \times B$ to bit $k \times B - 1$. $D(x, k)$ is also called the k-th "digit" of x. In addition, we define the level-k "prefix" of an address x to be a string composed of the first $k - 1$ digits of x, and denote it by $P(x, k)$. Note that any level-1 prefix is a null string. For a node u, $D(A(u), k)$ is briefly written as $D(u, k)$, and $P(A(u), k)$ as $P(u, k)$. Likewise, for a packet q, $D(A(q), k)$ is briefly written as $D(q, k)$, and $P(A(q), k)$ as $P(q, k)$.

IV. TUNNEL VECTOR (TV) ALGORITHM

Our design of the Tunnel Vector (TV) algorithm is motivated by a DHT approach previously proposed in [9], and is described in more details in this section.

Each node u maintains a routing table that contains the TVs at multiple levels. A tunnel vector is in form of $\{k, t, v, c\}$ where k is the *routing level*, t is a *tunnel* that is an ordered sequence of the intermediate nodes on a path from u to the *end node* v, and c is the total *cost* of that path. Note that if t is null, i.e. contains no node, the path is directly from u to v with no intermediate node at level k. Node u also maintains variables $F(k, g)$ and $C_{min}(k, g)$ for each level k and each digit $g \in \{0, 1, 2 \ldots 2^B - 1\}$, where $F(k, g)$ records the end node v of the TV with the minimal cost c for the (k, g) pair, and c is recorded by $C_{min}(k, g)$. It consistently assigns the correct values to $C_{min}(k, g)$ and $F(k, g)$ dynamically according to the TVs recorded in the routing table. Note that the (k, v) pair of a TV is unique in the routing table. Thus, a TV update will overwrite an old TV with the same (k, v) pair. If and only if a physical link $(x, y) \in E$, x records a TV $\{1, \varepsilon, A(y), C(x, y)\}$ and y records a TV $\{1, \varepsilon, A(x), C(y, x)\}$, where ε is a null tunnel.

Each packet q has a packet header that contains the destination address $A(q)$, an integer variable $K(q)$ for the

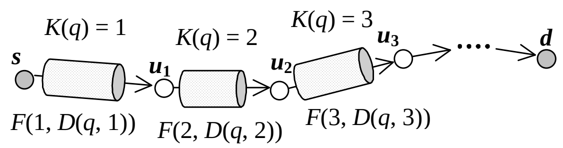

Figure 3 – The forwarding chain of a packet *q*

```
PROCEDURE Forward (packet $q$)
BEGIN

  IF $|S(q)| = 0$, THEN
  BEGIN /* Routing the packet at the higher level. */

    Increase $K(q)$ by 1.
    Find the TV $\{k, t, v, c\}$
      such that $k = K(q)$ and $v = F(k, D(q, k))$.
    IF no such TV exists OR $P(v, k) \neq P(q, k)$,
      THEN Discard $q$ and return;
      ELSE BEGIN
        LET a path $p \leftarrow t$ appended with $v$.
        Push $p$ into $S(q)$. /* Tunneling at level $K(q)$. */
      END

  END

  Pop a path $s$ from $S(q)$.
  LET an address $f \leftarrow$ the first node address in $s$.
  Remove $f$ from $s$.

  WHILE $|S(q)| + 1 < K(q)$, DO
  BEGIN /* Recursively tunneling until level 1. */

    Push $s$ into $S(q)$.
    Find the TV $\{k, t, v, c\}$
      such that $k = K(q) - |S(q)|$ and $v = f$.
    LET $s \leftarrow t$ appended with $v$.
    LET $f \leftarrow$ the first node address in $s$.
    Remove $f$ from $s$.

  END

  WHILE $s$ is a null path AND $|S(q)| > 0$, DO
  BEGIN /* Discarding null paths at tunnel ends. */
    Pop a path $r$ from $S(q)$.
    LET $s \leftarrow r$.
  END

  IF $s$ is NOT a null path,
    THEN push $s$ into $S(q)$.

  Send $q$ directly to the node at address $f$.

END
```

Figure 4 – The pseudo code of "Forward"

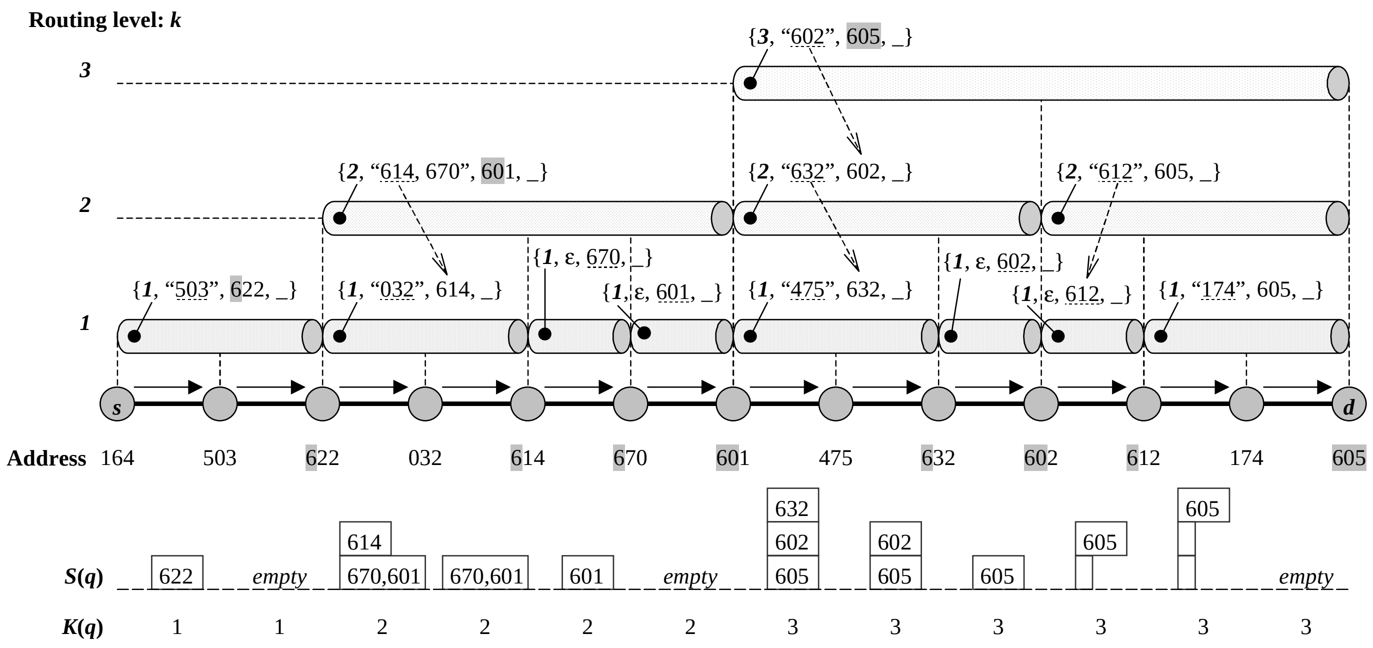

Figure 5 – An example of forwarding a packet q with $A(q)$ = 605 from a node s with $A(s)$ = 164

current "routing level," and a "source-routing stack" $S(q)$. The *stack* is a data structure that supports two primitive operations *push* and *pop*. Items can be pushed into $S(q)$ and popped out in the order of "last in, first out." The size of $S(q)$ is the number of the items remaining in it, and is denoted by $|S(q)|$. When q initially enters the network, $K(q)$ is reset to zero and $S(q)$ empty, unless otherwise specified.

A. *Forwarding a Packet*

Referring to Figure 3, a packet q is forwarded in a chain composed of tunnels. Through the k-th tunnel, q is forwarded to an end node u_k at address $F(k, D(q, k))$ with the routing level $K(q) = k$. The mechanism is in fact adapted from the routing principle of the PRR [12] scheme that is employed by DHT proposals such as Tapestry and Pastry, and can provide a relatively short latency with high probability, i.e. with a $O(1)$ ratio of the 'achieved' latency to the shortest-possible latency on the shortest path. In Figure 4, the pseudo-code for the forwarding chain is listed in the inner bounding box in the "Forward" procedure.

Semantically, a TV $\{k, t, v, c\}$ indicates a tunnel in which a packet q with $P(q, k + 1) = P(v, k + 1)$ can be forwarded to v. If $k > 1$, each hop in the tunnel, say, from a node x to the next node y, can refer to a non-existing link in the network, i.e., the physical link $(x, y) \notin E$. As previously mentioned (see Figure 1), TVs at level $k – 1$ are used to forward q from x to y over that hop. The recursive tunneling is performed by the pseudo code in Figure 4 where statements in boldface indicate the primary *push* and *pop* operations on $S(q)$ and how to assign variable f. Each item in $S(q)$ refers to a source-routing path of a tunnel that q is currently traversing at a unique level, and the source-routing path contains only the remaining nodes for q to visit in that tunnel. At the end of the "Forward" procedure, q is directly sent to the next node at address f.

Figure 5 shows an example of forwarding a packet q from the source node s to the destination node d, with $A(q) = A(d) = 605$, $B = 3$ and $L = 3$. The addresses are shown in radix 8 (= 2^3). The associated TV of each tunnel is shown but the cost of the TV is omitted. The forwarding chain of q is composed of three tunnels at routing level 1, 2, and 3, respectively. At the source node s, a TV {1, "503", 622, _} is referenced, and s increases $K(q)$ to 1 and pushes "622" into $S(q)$ before it forwards q to node 503. Node 503 pops "622" from $S(q)$ and sends q to node 622 with empty $S(q)$. Since $S(q)$ is found empty by node 622, $K(q)$ is increased to 2. With a TV {2, "614, 670", 601, _}, node 622 starts the second tunnel in the forwarding chain and assigns f = 614. Then, it seeks for a TV that has the end node equal to f, and gets TV {1, "032", 614, _}. It pushes "670, 601" into $S(q)$, and reassigns f = 032. Now that $|S(q)| + 1 = K(q)$, it knows the recursive tunneling has reached level 1. Thus, it pushes "614" into $S(q)$, and forwards q to address f = 032. The remainder of the routing proceeds similarly. The underlined addresses are those that have been assigned to variable f at each forwarding node. Finally, at the end of the third tunnel, q reaches node 605.

B. *Exchanging the TVs*

Initially, the routing table at each node x contains no TV, and thus, for each digit $g \in \{0, 1, 2, ... 2^B – 1\}$ at each level k, $C_{min}(k, g) = \infty$ and $F(k, g)$ is null. Then, x records a TV $\{k, ε, A(x), 0\}$ for each $k = 1, 2, ... L$ since $C(x, x) = 0$, and records a

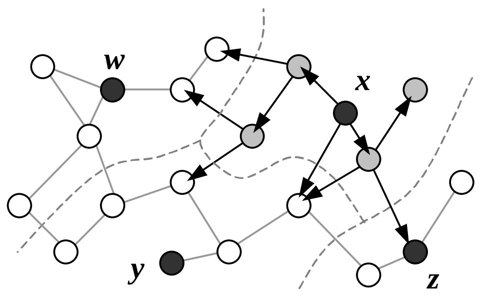

Figure 6 – "Advertise" messages with TV {*k*, _, *A*(*x*), _}

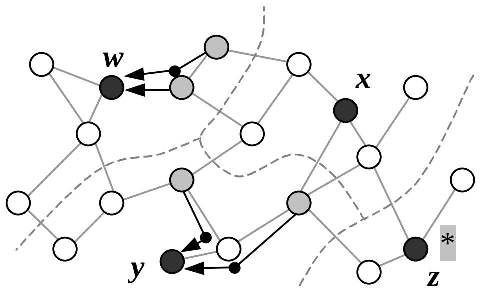

Figure 7 – "Acquaint" messages with TV {*k*, _, *A*(*x*), _}

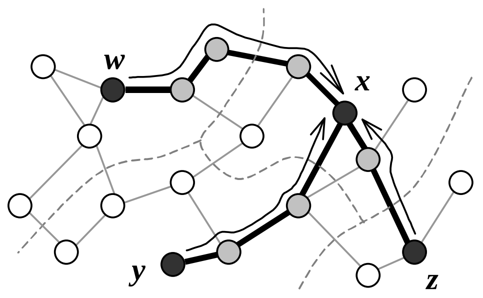

Figure 8 – Generated tunnels with an end at *x*

TV {1, ε, *A*(*y*), *C*(*x*, *y*)} for each node *y* such that there is a physical link (*x*, *y*) ∈ *E*.

Referring to Figure 6, the NID of each node shares the same level-*k* prefix, and *D*(*w*, *k*) = *D*(*x*, *k*) = *D*(*y*, *k*) = *D*(*z*, *k*). In general, if *F*(*k*, *g*) at a node *u* is non-null for a level *k* and a digit *g*, then *u* generates an "Advertise" message with a recorded TV {*k*, *t*, *F*(*k*, *g*), C_{min}(*k*, *g*)} and places *A*(*u*) at the beginning of *t* in the message. Node *u* then sends this message to all its *neighbors* at level *k*, where a "neighbor" at level *k* is defined to be a node *v* (≠ *u*) such that there exists a TV {*k*, ε, *v*, _} in the routing table of *u*. However, if either *A*(*v*) = *F*(*k*, *g*) or if *A*(*v*) appears in *t*, *u* does not send *v* the Advertise message since that would cause a loop. To send an Advertise message, *u* generates a packet *q* with *A*(*q*) = *v*, *K*(*q*) = *k*, and empty *S*(*q*), and pushes "*A*(*v*)" into *S*(*q*) before it calls the "Forward" procedure. Then, as *q* traverses the network, any node *r* that receives *q* directly from a node *s* retrieves the TV {*k*, *t*, *A*(*x*), *c*} in *q*, and lets *c* ← *c* + *C*(*r*, *s*). If *A*(*q*) ≠ *A*(*r*), *r* simply calls the "Forward" procedure. Otherwise, *r* records the updated TV in its routing table. If the update changes the value of *F*(*k*, *g*) or C_{min}(*k*, *g*) for *g* = *D*(*x*, *k*), *r* generates a new Advertise message to its neighbors in a similar way. Otherwise, *r* will generate an "Acquaint" message to *F*(*k*, *g*) if *F*(*k*, *g*) ≠ *A*(*x*). For example, in Figure 7, the gray nodes generate Acquaint messages to either *w* or *y* after receiving an Advertise message with a TV {*k*, _, *A*(*x*), _} in Figure 6.

The Acquaint message generated by *r* has a TV {*k*, *t′*, *A*(*x*), *c*} where *t′* displays a path from *F*(*k*, *g*) to *A*(*x*). Refer to Figure 8. To computes *t′*, *r* finds a recorded TV {*k*, *t**, *F*(*k*, *g*), _} and lets t′ ← *R*(*t**) • "*A*(*r*)" • *t* where *R*(*t**) is the reverse path of *t**, and '•' is an operator to concatenate two paths. The node at *F*(*k*, *g*), e.g. *w*, records the TV and creates a new TV {*k* + 1, ε, *A*(*x*), *c*} in its routing table so that *x* becomes its "neighbor" at level *k* + 1. However, if *F*(*k*, *g*) = *A*(*r*), e.g. at *z*, it implicitly (marked with an asterisk) makes the new TV {*k* + 1, ε, *A*(*x*), *c*} at *r*. For an Acquaint message, a packet is generated by *r* and forwarded by nodes in the same way as for an Advertise message.

If a new physical link is created, the linked nodes are triggered to send Advertise messages to each other at level 1. The other necessary messages at higher levels will accordingly be sent. If a node updates the cost of a physical link, it updates all associated TVs and then sends Advertise messages that will cascade.

V. Conclusion and Future Work

We have identified the major challenges of adapting a DHT method to a network-layer routing algorithm, and proposed a new network-layer routing algorithm. With recursively tunneling, it uses only physical links, handles linkage updates, and can provide high routing efficiency with high probability. With an average size *O* (log *n*) of its routing tables, TV achieves excellent scalability and mobility while the latency in routing a packet is near optimal with high probability.

In the future work, it will be necessary to further study the performance, search for possible improvement and limitation, and find support for multicast, anycast [1], QoS, policy routing [5][20], etc., for practical uses in large global networks.

Note that since the proposed TV routing algorithm takes into account network distances when building its conceptual overlay links, we argue that with carefully designed metrics of network distances, it is possible to deploy TV in conjunction with hierarchical routing, for maximal flexibility in mobility and efficiency. It would potentially allow simple migration of the routing software/firmware from the current hierarchical routing scheme to a generalized TV-based routing scheme that can cover the global network with no limitation of hierarchy of host addresses or subnetworks, and easy migration is typically considered essential for practice uses.

References

[1] H. Ballani, P. Francis. Towards a global IP anycast service. In *Proc. of ACM SIGCOMM Conf.*, 2005.

[2] S. Deering, R. Hinden. Internet protocol, version 6 (IPv6) specification. *IETF RFC* 2460, Dec 1998.

[3] J. Eriksson, M. Faloutsos, S. Krishnamurthy. PeerNet: Pushing peer-to-peer down the stack. In *Proc. of the 2nd Int'l Workshop on P2P Systems*, Feb 2003.

[4] B. Ford. Unmanaged Internet Protocol: Taming the edge network management crisis. In *Proc. of ACM HotNet-II Workshop*, 2003.

[5] N. Feamster, R. Johari, H. Balakrishnan. Implications of Autonomy for the Expressiveness of Policy Routing. In *Proc. of ACM SIGCOMM Conf.*, 2005.

[6] J.J. Garcia-Luna-Aceves. Loop-free routing using diffusing computations. In *IEEE/ACM Transactions on Networking*, Vol. 1, No. 1, Feb 1993.

[7] N. J. A. Harvey, M. B. Jones, M. Theimer, A. Wolman. Efficient recovery from organizational disconnects in SkipNet. In *Proc. of the 2nd Int'l Workshop on P2P Systems*, Feb 2003.

[8] K. Hildrum, J. D. Kubiatowicz, S. Rao, B. Y. Zhao. Distributed object location in a dynamic network. In *Proc. of ACM SPAA*, 2002.

[9] C.J. Lai, R. R. Muntz. On-demand overlay networking of collaborative applications. In *Proc. of the 1st IEEE Collaborative Computing Conf. (CollaborateCom 2005)*, Dec 2005.

[10] J. Moy. OSPF version 2. *IETF RFC* 2328, Apr 1998.

[11] V. Paxson. End-to-end routing behavior in the Internet. In *IEEE/ACM Trans. on Networking*, Vol. 5, No. 5, Oct 1997.

[12] G. Plaxton, R. Rajaraman, A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *Proc. of ACM SPAA*, Jun 1997.

[13] V. Ramasubramanian, E. G. Sirer. Beehive: Exploiting power law query distributions for O(1) lookup performance in peer to peer overlays. In *Proc. of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI '04)*, Mar 2004

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

[15] Y. Rekhter, T. Li. A Border Gateway Protocol 4 (BGP-4). *IETF RFC* 1771, Mar 1995.

[16] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, H. Yu. OpenDHT: A public DHT service and its uses. In *Proc. of ACM SIGCOMM Conf.*, 2005.

[17] A. Rowstron, P. Druschel. Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proc. of IFIP/ACM Int'l Conf. On Distributed Systems Platforms*, Nov 2001.

[18] J. L. Sobrinho. Network routing with path vector protocols: theory and applications. In *Proc. of ACM SIGCOMM Conf.*, 2003.

[19] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for Internet applications. In *Proc. of ACM SIGCOMM Conf.*, 2001.

[20] L. Subramanian, M. Caesar, C.T. Ee, M. Handley, M. Mao, S. Shenker, I. Stoica. HLP: A next generation inter-domain routing protocol. In *Proc. of ACM SIGCOMM Conf.*, 2005.

[21] B. Y. Zhao, L. Huang, A. D. Joseph, J. Kubiatowicz. Rapid mobility via type indirection. In *Proc. of the 3rd Int'l Workshop on P2P Systems*, Feb 2004.

[22] L. Zhou, R. Renesse. P6P: A peer-to-peer approach to Internet infrastructure. In *Proc. of the 3rd Int'l Workshop on P2P Systems*, Feb 2004.