Title

THORP: Choosing Ordered Neighbors To Attain Efficient Loop-Free Minimum-Hop Routing

Author

Garcia-Luna-Aceves, J.J.

Publication Date

2022-07-01

Data Availability

The data associated with this publication are within the manuscript.

Peer reviewed

THORP: Choosing Ordered Neighbors To Attain Efficient Loop-Free Minimum-Hop Routing

J.J. Garcia-Luna-Aceves
Computer Science and Engineering Department
University of California, Santa Cruz
Santa Cruz, CA, USA
jj@soe.ucsc.edu

***Abstract*—We introduce THORP (Totally Hop-Ordered Routing Procedure), a simple distributed algorithm for minimum-hop routing that works in much the same way as traditional distance-vector routing algorithms do. THORP eliminates routing-table loops by having routers choose as their next hops to destinations those neighbor routers that are totally ordered based on their current distances, without requiring their next-hop routers to correspond necessarily to minimum-hop paths. THORP is shown to be loop-free, to converge to minimum-hop distances within a finite time, and to be faster than the Diffusing Update Algorithm (DUAL), which is the only loop-free shortest-path algorithm that has been used successfully in practice and is part of Cisco's EIGRP.**

***Index Terms*—loop-free routing, minimum-hop routing, multi-path routing**

I. Introduction

Most of the routing algorithms and protocols that run in Internet, wireless networks, and the IoT today are based on different approaches to shortest-path routing, which is the subject of this paper. The Distributed Bellman-Ford (DBF) algorithm is arguably the simplest example of these protocols, and has been used in many routing protocols in the past, including the original ARPANET routing protocol [15], the Routing Information Protocol (RIP) [9], and RIPv2 [11].

As Section II summarizes, many approaches have been proposed to eliminate the well-known looping problems associated with DBF. These approaches are based on either destination sequence numbers that identify which distance data are more recent, or inter-nodal coordination carried out while routing tables are updated. As the examples in Section III show, current approaches to loop-free routing based on distance information are not well-suited for networks in which minimum-hop routing is used, because they tend to over-react to topology changes in their attempt to seek shortest paths at the same time that loop freedom is maintained.

This paper introduces a different approach to distributed minimum-hop loop-free routing in computer networks. Rather than having routers search for minimum-hop paths while attempting to prevent routing-table loops, each router selects as its next hops to a destination those neighbor routers that report shorter distances than the distance currently maintained by the router itself, even if the choices do not correspond to perceived minimum-hop paths.

Sections IV and V present the *Totally Hop-Ordered Routing Procedure* (THORP), which provides one or multiple loop-free minimum-hop routes to destinations at each router if they exist, without incurring long delays converging to such routes. In a nutshell, a router can change its next hop to a destination only if it has neighbor routers with reported distances that are strictly smaller than its own distance. Otherwise, the router sends a query stating its current distance and a reference distance equal to the value of its own distance prior to the input event that prompted the query. A router that receives a query and knows of a neighbor with distance shorter than the reference distance stated in the query sends a reply stating its own distance and the reference distance in the query; otherwise, the router propagates the query specifying its own distance and the requested distance in the query it received. A router that forwards a reply states its own distance and the reference distance in the response it receives.

Section VI shows that THORP is loop-free and converges to minimum-hop routes within a finite time.

Section VII shows that THORP converges to valid routes faster than DUAL [7], which has been used successfully for many years as part of Cisco's EIGRP [14].

Section VIII provides our conclusions.

II. Related Work

DBF suffers from the non-convergence problem usually called the counting-to-infinity problem. As a result, routing protocols based on DBF are forced to declare a destination to be unreachable when a predefined maximum-distance value is reached, which limits the type of routing metric that can be used, induces long convergence delays, and need not result in convergence to shortest paths.

Over the years, many routing algorithms have been developed to eliminate the convergence problems of DBF, and in some cases also prevent temporary loops. The first example of alternatives to DBF was the broadcasting of complete topology information together with the local use of Dijkstra's shortest-path first (SPF) at each router [15]. Many routing protocols and algorithms (e.g., [8], [12]) are based on this approach and are known to provide fast convergence on average, even though they do not prevent temporary routing loops. Other approaches

978-1-6654-8353-7/22/$31.00 (c)2022 IEEE

use partial topology information in the form of link states (e.g., [2], [16]), distances and second-to-last hops to destinations (e.g., [3]), or complete path information in updates (e.g., DSR [10]) and use different shortest-path algorithms locally.

A number of shortest-path routing protocols (e.g., DSDV [13]) use destination sequence numbers to attain loop freedom, and RPL (Routing Protocol for Low Power and Lossy Networks) [5], [20] is the most recent example. Even though using sequence numbers to provide acyclic operation is appealing, the basic approach has been shown to incur temporary routing-table loops when router failures, volatile memory, and recycling of sequence numbers are involved (e.g., see [17]).

Several shortest-path routing approaches have been developed that provide loop-free routing by requiring nodes to coordinate the updating of routing tables on a multi-hop basis. However, the most popular of these schemes is the Diffusing Update Algorithm (DUAL) [7] and is the basis of Cisco's EIGRP [1], [14]. Subsequent protocols apply this approach to attain acyclic multi-path routing based on link-state information [18] or distance vectors [6], [19], [21]. Babel [4] is a recent proposal for acyclic routing based on combining the diffusing computations introduced in DUAL with sequence numbering.

Many other routing protocols have been proposed, but due to space limitations we do not discuss them. DUAL is the only loop-free routing algorithm that has been used successfully for many years as part of a routing protocol, specifically EIGRP, and hence we use it for comparison purposes.

III. Motivation for THORP

Figure 1 shows a seven-node network in which DBF is executed and link (c, d) fails. Nodes are routers and destinations, and the distance from each router to destination d is stated next to each router and arrowheads indicate the next hop to d. Updates are indicated by $U(h)$, where h is the hop count to destination d.

For convenience, it is assumed that the network operates synchronously in steps. Each router processes all messages it received in the previous step and sends an update with the result in the current step. As the figure illustrates, DBF converges in just three steps in this scenario, and only a short-lived routing-table loop is formed between routers b and c.

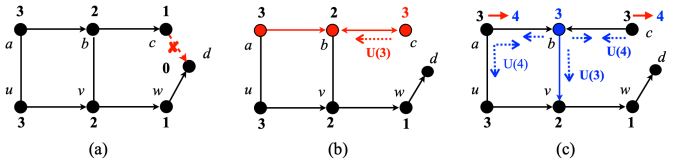

Fig. 1. Quick convergence with looping in DBF

Interestingly, the same short-lived routing loop would occur using the topology-broadcast method, and the dissemination of link-state information would take more steps than the steps needed for DBF to converge. Unfortunately, even though DBF can be very efficient in many cases, it does not always perform well and cannot converge to correct routes after destination failures or network partitions. For example, counting-to-infinity would occur if both links (c, d) and (w, d) were to fail. Furthermore, more complex routing loops could occur in different topologies.

DUAL has been used successfully in EIGRP for quite some time to eliminate routing loops. However, DUAL may result in multiple routers being blocked from having next hops to destinations while routers recompute shortest paths. Figure 2 illustrates this problem using the scenario of Figure 1.

DUAL requires a router to trust a distance reported from a neighbor only if it renders a shortest distance and is strictly smaller than its own *feasible distance* for a destination. The feasible distance for a destination is the smallest value of the local distance before a router sends a query. Let h_d^a and f_d^a be the distance and feasible distance at router a for destination d, respectively, and N^a be the set of neighbors of router a. The "source node condition" (SNC) [7] imposed on a router a to select a successor q for destination d is:

$$\mathcal{SNC} : (h_d^q < f_d^a) \wedge (h_d^q = Min\{h_d^n + l(a,n) \mid n \in N^a\})$$

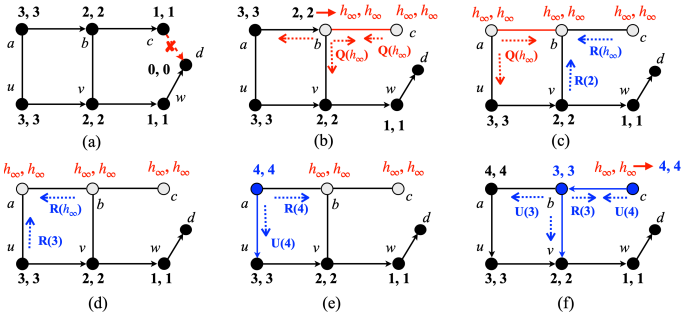

Fig. 2. Delayed convergence and blocking in DUAL

As long as router a has a neighbor that satisfies SNC, it updates its distance and next hop as in DBF and accordingly sets $f_d^a = Min\{f_d^a, h_d^a\}$ when it changes h_d^a.

If SNC is not satisfied and minimum-hop routing is used, router a sets $h_d^a = h_\infty$ and $f_d^k = h_\infty$, and sends a query stating $h_d^a = h_\infty$ to all its neighbors (denoted by $Q(h_\infty)$ in Figure 2). Router a then must wait for replies (denoted by $R(h)$ in Figure 2) from all its neighbors before it is allowed to change its next hop for destination d.

A router sends a reply to a query when: (a) it has already sent a query, (b) its own distance is not affected by the query, or (c) it receives all the replies to a query that it forwarded or originated. Because of this type of signaling, DUAL may incur long delays to converge and may cause many routers to be blocked while queries and replies propagate along the directed acyclic graphs (DAG) defined by the next-hop entries in routing tables.

In the example of Figure 2, it takes seven steps for routers a, b, and c to attain valid paths while the multi-hop coordination is taking place, even though physical paths to destination d do exist. This motivates the need to explore a more efficient approach to avoid routing loops in the context of minimum-hop routing. The intended result is a distributed routing algorithm that is loop-free, is as fast as DBF in the absence of network partitions or destination failures, and converges quickly when network partitions or destination failures occur.

IV. THORP

A. Overview

THORP uses routing messages that contain updates, queries, and replies. An update states the current minimum-hop distance at the router sending the update. A query specifies the current distance to a destination and a reference distance that must be met for a response to be sent. A reply states the distance of the router sending the reply and the reference distance stated in the query that prompted the reply.

The reference distance used in THORP is similar in some sense to the feasible distance used in DUAL. As long as a router can choose as its next hop to a destination a neighbor that reported a distance that is shorter than its own reference distance, the router updates its routing table and sends updates in much the same way as in DBF, and its distance and reference distance are the same. However, if a router cannot find a neighbor with a reported distance strictly smaller than its reference distance, then the router remembers the value of its reference distance, updates the value of its own distance as needed, and sends a query to all its neighbors stating its distance *and* reference distance. The query is intended to find a router with a distance that is shorter than the reference distance it states.

The signaling of both THORP and DUAL include queries and replies; however, they use them differently. A router in DUAL resets its feasible distance to equal infinity before it sends a query, and a query only states the current distance of a router. This allows a router to trust a new minimum distance when all neighbors send their replies, but prevents the router from being able to trust the first valid reply. By contrast, THORP allows a router to trust the first reply to its query by making routers remember their reference-distance values and by including those values in queries and replies. Furthermore, storing reference distances enables routers to forward queries to their next hops to destinations if they have valid routes and are simply helping to resolve a query.

A query from a router p propagates until it reaches a router r that has a neighbor n with a distance smaller than the reference distance stated in the query. Router r originates a reply stating its current distance plus the reference distance in the query. This causes replies to be sent back towards router p that state the same reference distance reported by router r and the current distances at the relaying routers.

A router that initializes or restarts cannot send or process new routing messages until an initialization delay elapses that is longer than the time needed by all its prior neighbors to determine that the router stopped being their neighbor. This is done simply to tolerate the loss of routing state due to router failures and he lack of non-volatile memory.

B. Information Exchanged and Maintained

The set of network nodes (routers and destinations) is denoted by N, and E denotes the set of links in a network. A node in N is denoted by a lower-case letter, and a link between nodes n and m in N is denoted by (n, m). The set of nodes that are immediate neighbor routers of router k is denoted by N^k.

Routers may maintain multiple routes to destinations. The path corresponding to the nth route at router k to destination d is denoted by $P_d^k(n)$, and the next hop along that path is denoted by $s_d^k(n)$, with $s_d^k(n) \in N^k$. Path $P_d^k(n)$ consists of the concatenation of the link $(k, s_d^k(n))$ with a path $P_d^{s_d^k(n)}(m)$, i.e., $P_d^k(n) = (k, s_d^k(n))P_d^{s_d^k(n)}(m)$.

Routers exchange routing messages reliably to update their routing information. A routing message from router k is denoted by M^k and contains its identifier k and a list of one or more entries, each with an update, query, or reply.

An update for destination d is denoted by $U(d, h_d^k)$, where h_d^k is the local minimum-hop distance to destination d.

Update $U(k, 0)$ from router k indicates that the update is a "hello" to refresh the presence of k.

A query is denoted by $Q(d, h_d^k, \rho_d^k)$, where ρ_d^k is a requested distance equal to the smallest requested distance received or computed at the router propagating the query.

A reply is denoted by $R(d, h_d^k, \rho_d^k)$, where ρ_d^k is copied from the query being answered.

Each router k knows its own identifier (k) and maintains a Neighbor Table (NT^k), a Distance Table (DT^k), a Routing Table (RT^k), and its initialization status (σ^k).

NT^k lists an entry for each known neighbor router $n \in N^k$. The entry for neighbor n states a lifetime LT_n^k for the neighbor entry of router n. The maximum lifetime of a neighbor entry is a constant LT defined for the network.

DT^k lists the information reported by each neighbor. The entry in DT^k for destination d at router k is denoted by $DT^k(d)$ and specifies for each neighbor $p \in N^k$ the distance h_{dp}^k to destination d. If a neighbor q has not reported any distance for d, then router k sets $h_{dq}^k \leftarrow h_\infty$ and $o_{dp}^k \leftarrow 0$.

RT^k lists an entry for each destination d. The entry in RT^k for destination d is denoted by $RT^k(d)$ and states: The minimum-hop distance (h_d^k), the preferred next hop (s_d^k), the set of next hops (S_d^k), and a local reference distance (r_d^k). Setting $s_d^k = 0$ when $S_d^k = \emptyset$ denotes the fact that there is no next hop to d.

The value of the reference distance r_d^k equals the value of h_d^k when the router has one or more valid next hops, or the smallest value of a requested distance stated in a query created or forwarded by the router when it becomes active.

C. Signaling

For simplicity, it is assumed that routing messages are sent reliably, such that a router receives an acknowledgment to its routing message from a neighbor before it sends the next message to the same neighbor. Messages are sent periodically after routers wait for short or long time intervals.

Initialization: A router k is initialized after an initialization delay elapses. At that point, the router has a neighbor set N^k, $\sigma^k = T$, and sets $h_k^k = r_k^k = 0$, and $s_k^k = k$. For each $q \in N^k$, the router also sets $h_{kq}^k = r_{kq}^k = h_\infty$, $h_{qq}^k = r_{qq}^k = h_\infty$. Router k then sends a routing message with $U(k, 0)$.

Periodic Messaging: Router k maintains a timer UT^k to ensure that it sends a routing message soon after it updates its routing table or decides to forward or respond to a query, and sends routing messages often enough to inform its neighbors of its presence. If a router k needs to send a routing message with updates, it does so after a minimum amount of time t_{min} has elapsed from the time it sent its prior routing message.

In the absence of updates to its routing table, a router sends a message with a "hello" update $U(k,0)$ to update the lifetime entries maintained for itself by its neighbors no later than t_{max} seconds from the time it sent its last message. The timer t_{max} is shorter than a maximum lifetime LT. Router k sets UT^k equal to t_{max} after sending a routing message, and sets UT^k equal to t_{min} after preparing updates or queries to be sent in response to an input event.

D. Updating NT^k and DT^k

Router k updates NT^k when an adjacent link (k,q) changes status, and updates DT^k when any type of input event occurs, such as when an adjacent outgoing link changes its weight, an immediate neighbor router fails to send updates before the lifetime for its entry in NT^k expires, or a routing message is received from a neighbor.

A router k detects that a new neighbor q is present when it receives a "hello" update from q ($U(q,0)$) and its local state for q has $h^k_{qq} = h^k_{kq} = h_\infty$, which indicates that no messages were being received over link (k,q). In this case, router k immediately sends a routing message with an update $U(d,h^k_d)$ for each destination d for which $h^k_d < h_\infty$. If router k receives an update, a query, or a reply from neighbor q reporting a distance h^q_d, then it updates $h^k_{dq} \leftarrow h^q_d$.

E. Updating RT^k

After router k updates NT^k and DT^k, it updates RT^k as part of the processing of an input event as shown in Algorithm 1. The way in which router k updates RT^k depends on its routing state, which is determined by the following condition $\mathcal{T}$:

$$\mathcal{T}: \left(\exists\ q \in N^k \left[r^k_d > h^k_{dq}\right]\right) \vee \left(\forall q \in N^k \left[h^k_{dq} = h_\infty\right]\right) \quad (1)$$

Condition $\mathcal{T}$ states that router k either has neighbors with reported distances smaller than its own reference distance, or all its neighbors have declared the destination to be unreachable. A router is said to be passive if $\mathcal{T}$ is true and is active otherwise.

We observe from Algorithm 1 that $r^k_d = h^k_d$ as long as router k is passive. Accordingly, router k can determine that it is passive if $r^k_d = h^k_d$ and that it is active otherwise. Router k sends an update, a query, or a reply depending on the the input event and whether it is passive or active.

Processing Updates: Router k takes the following steps when it receives update $U(d,h^q_d)$ or detects a link-status change:

(a) If router k remains or becomes passive, then it sends an update $U(d,h^k_d)$.

(b) If router k becomes active, then router k originates a query $Q(d,h^k_d,\rho^k_d = r^k_d)$.

(c) If router k remains active after the input event and at least one neighbor v has reported a finite distance, then router k sends a query $Q(d,h^k_d,\rho^k_d = r^k_d)$ if h^k_d was updated; otherwise, router k stays silent.

Processing Replies: Router k takes the following steps when it receives reply $R(d,h^q_d,\rho^q_d)$ from neighbor q:

(a) Router k sends a reply $R(d,h^k_d,\rho^k_d = \rho^q_d)$ if it either becomes passive and $\rho^q_d \le r^k_d$, or it remains passive and either $\rho^q_d < r^k_d$ or the value of h^k_d was updated. These actions help propagate a reply towards the origin of a pending query or provide a necessary update to its neighbors.

(b) Router k originates a query $Q(d,h^k_d,\rho^k_d = r^k_d)$ if it becomes active as a result of the reply from neighbor q. However, if q was in S^k_d before the reply made router k become active, the path from q to d cannot include k because $\mathcal{T}$ is true at q. Accordingly, router k updates $s^k_d \leftarrow q$, $S^k_d \leftarrow \{q\}$, $h^k_d \leftarrow h^q_d$.

(c) Router k stays silent if it was active before it processes the reply from q and remains active after it processes the reply.

Processing Queries: If router $k$receives query $Q(d,h^q_d,\rho^q_d)$ from neighbor q, the router takes the following steps based on its own state and the content in the query:

(a) Router k sends reply $R(d,h^k_d,\rho^k_d = \rho^q_d)$ if it is passive and has a neighbor v such that $h^k_{dv} \le \rho^q_d$.

(b) Router k sends query $Q(d,h^k_d,\rho^k_d = \rho^q_d)$ if it remains passive but it has no neighbor v such that $h^k_{dv} \le \rho^q_d$.

(c) Router k sends query $Q(d,h^k_d,\rho^k_d = \rho^q_d)$ if it either becomes active or remains active and $\rho^q_d < r^k_d$. It also updates $r^k_d \leftarrow Min\{r^k_d,\rho^q_d\}$.

(d) Router k stays silent if it is active before the query from q is received and all its neighbors have reported h_∞.

Algorithm 1 Routing Table Update ($RT^k(d)$)

INPUT: NT^k, $DT^k(d)$, $RT^k(d)$
ROUTER IS PASSIVE:
if $\left(\forall q \in NT^k(h^k_{dq} = h_\infty)\right)$ **then**
  Destination is unreachable:
  $S^k_d \leftarrow \emptyset$; $s^k_d \leftarrow 0$; $h^k_d \leftarrow h_\infty$
end if
if $\left(\exists q \in NT^k \left([h^k_{dq} < r^k_d]\right)\right)$ **then**
  Router k is ordered with respect to d:
  $h^k_d \leftarrow Min\{h^k_{dn} + 1 \mid n \in NT^k\}$; $r^k_d \leftarrow h^k_d$;
  $S^k_d \leftarrow \{n \in NT^k \mid h^k_{dn} = h^k_d - 1\}$;
  $s^k_d \leftarrow Min\{q \in S^k_d\}$
end if
ROUTER IS ACTIVE:
if $\left(\forall q \in NT^k \left(h^k_{dq} \ge r^k_d\right)\right)$ **then**
  $h^k_d \leftarrow h_\infty$; $r^k_d \leftarrow r^k_d$ (reference distance is not updated);
  $S^k_d \leftarrow \emptyset$; $s^k_d \leftarrow 0$
end if

V. Examples of THORP Operation

Figure 3 illustrates why THORP provides loop-free paths faster than DUAL using the same example of Figures 1 and 2. The distance and reference distance for destination d are indicated next to each router. Next hops to destinations are

indicated by arrowheads. An update, query, and response sent by router k for destination d is denoted by $U[h_d^k]$, $Q[h_d^k, \rho_d^k]$, and $R[h_d^k, \rho_d^k]$, respectively.

Routers a, b, and c are blocked as in DUAL; however, blocking lasts for only five steps rather than seven, because an active router can become passive with the first reply to its query. Router c becomes active and sends a query $Q(d, h_\infty, 1)$ when link (c, d) fails, and routers b and c become active and send queries when they receive queries with a distance of h_∞ from the only neighbors that had reported shorter distances to d than heir own distances. Router v replies to the query from b with $R(d, 2, 1)$, because $h_{dw}^v = 1 = \rho_d^b$. Router b becomes passive after processing the reply from router v, router c becomes passive after processing the reply from b, and router c becomes passive after receiving the reply from b.

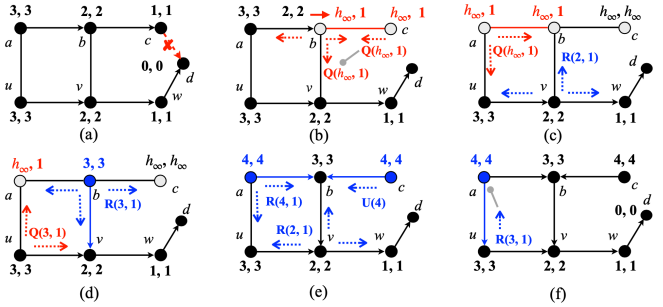

Fig. 3. Fast loop-free convergence in THORP

Figure 4 illustrates the loop-free and fast convergence of THORP after a network partition or destination failure. Routers take only four steps to converge to h_∞ without creating a loop.

DBF would count to infinity in this example. DUAL can be shown to require seven steps to converge, because the query started by router c must traverse the path $c \to b \to a \to u$ for replies to be sent to router c, and the query started by router w must traverse the path $w \to v \to u \to a$ for replies to be sent to router w.

The reader can also verify that four steps would be needed to attain consistent link-state information, but temporary routing loops would occur.

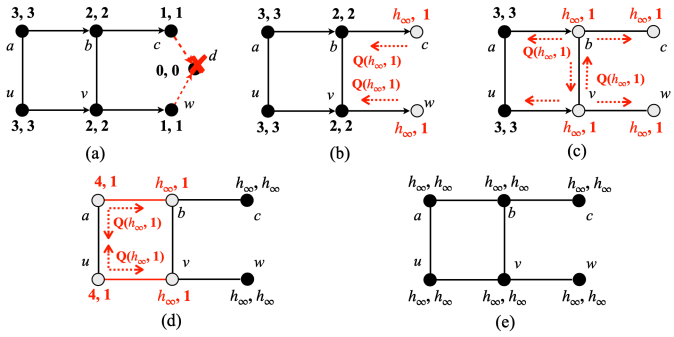

Fig. 4. Loop-freedom in THORP after a network partition or destination failure

VI. THORP Correctness

Theorems 1 to 6 prove that THORP is loop-free and converges to optimal paths within a finite time. The next hop, distance, and reference distance from router n to destination d at a given time t are denoted by $s_d^n(t)$, $h_d^n(t)$ and $r_d^n(t)$, respectively.

Theorem 1: A path in which $\mathcal{T}$ is satisfied at every router along the path cannot be a loop.

Proof: Assume that $\mathcal{T}$ is true at every router along a path L. For the sake of contradiction, assume that L is a routing-table loop that excludes destination d at time t and let $L = \{v_1 \to v_2 \to \ldots \to v_h \to v_{h+1}\}$, where $v_{h+1} = v_1$. Each router $v_i \in L$ informs its neighbors of its distance to d at a time denoted by t_i, where $t_i < t$, and its neighbors in L use that value at a subsequent time to determine whether $\mathcal{T}$ is satisfied. The time when router $v_i \in L$ makes router $v_{i+1} \in L$ a next hop to d is denoted by t_i^+ and $t_i^+ \leq t$, which implies that $s_d^{v_i}(t) = s_d^{v_i}(t_i^+)$, $h_d^{v_i}(t) = h_d^{v_i}(t_i^+)$, and $r_d^{v_i}(t) = r_d^{v_i}(t_i^+)$ for all $v_i \in L$.

The following results are a consequence of the fact that $\mathcal{T}$ must be satisfied at each router $v_i \in L$:

(a) $r_d^{v_i}(t_i^+) = r_d^{v_i}(t) > h_{dv_{i+1}}^{v_i}(t)$;

(b) $h_{dv_{i+1}}^{v_i}(t) = h_{dv_{i+1}}^{v_i}(t_i^+) = h_d^{v_{i+1}}(t_{i+1})$;

(c) $h_d^{v_{i+1}}(t_{i+1}) = r_d^{v_{i+1}}(t_{i+1}) = r_d^{v_{i+1}}(t)$.

It follows from (a), (b) and (c) that

$$\forall v_i \in L \ \left(r_d^{v_i}(t) > r_d^{v_{i+1}}(t) \right).$$

However, this constitutes a contradiction, because it implies that $r_d^{v_i}(t) > r_d^{v_i}(t)$ for all $v_i \in L$ and this is impossible; therefore, the theorem is true. ■

Theorem 2: No routing-table loop can be created in THORP when routers transition from passive to active state.

Proof: The proof follows from Algorithm 1, because a router that transitions to the active state either has no next hop or must keep its current next hop. The first case negates the existence of a routing-table loop. In the second case, the current next hop was part of a path established by routers in passive state, which negates the existence of a routing-table loop because of Theorem 1. ■

Theorem 3: THORP is loop-free for any destination d.

Proof: If $\mathcal{T}$ is always satisfied at every router, then it follows from Theorem 1 that no routing-table loops can form. Therefore, if follows from Theorems 2 that the proof needs to show that no routing-table loop can be created when a router transitions from active to passive state.

For a router k to become passive once it is active, it must receive an update or a response such that $\mathcal{T}$ is satisfied, and a router $n \in N^k$ can send an update or a response to router k only if it is passive itself.

The path from n to d either consists of routers that are passive, or consists of both active and passive routers. In the first case, it follows from Theorem 1 that router k cannot create a loop by setting $n = s_d^k$ because then the path from n to d is loop-free and extending that path with link (k, n)

cannot create a loop. In the second case, the path from n to d is the concatenation of subpaths, each consisting of one or more routers that are all passive or are all active, and it follows from Theorems 1 and 2 that such subpaths are loop-free and hence extending the path from n to d with link (k, n) cannot create a loop. Therefore, the theorem is true. ■

Theorem 4: THORP converges to minimum-hop routes for all reachable destinations within a finite time after network changes stop occurring in a finite network.

Proof: Assume that routers execute THORP correctly but router k converges incorrectly with $h_d^k > h_d^k[o]$, where $h_d^k[o]$ is the minimum-hop distance from router k to destination d according to the sum of link weights in any path from k to d.

Given that the network is connected and finite, all loop-free paths in the network are finite and it takes a finite time for updates and responses starting from destination d or any other router to propagate over any loop-free path. Furthermore, because THORP is executed correctly, the optimum distance $h_d^k[o]$ must correspond to a loop-free path from k to d through some neighbor router $s \in N^k$.

Updates, queries and replies must propagate over loop-free paths because THORP is loop-free (Theorem 3). This implies that router k must receive an update or a reply from router s reporting h_d^s, which makes $h_{ds}^k = h_d^s$. However, this is a contradiction to the assumption that THORP is executed correctly, because then Algorithm 1 would require router k to make $h_d^k[o] = 1 + h_{ds}^k$ its distance to destination d. ■

Theorem 5: THORP converges to h_∞ for all unreachable destinations within a finite time after network changes stop occurring in a finite network.

Proof: Assume that a router n_k belongs to a connected component from which destination d is unreachable starting at time t_0, and that no network changes occur after time $t_n \geq t_0$. For the sake of contradiction, assume that router n_k converges to a finite distance for d at time $t_k \geq t_n$.

If router n_k that has a distance $h_d^{n_k} < h_\infty$ at time t_k then it must have a next-hop neighbor $n_{k-1} \in N^{n_k}$ such that $h_{dn_{k-1}}^{n_k} < r_d^{n_k}$. Because THORP is loop-free (Theorem 3), there must be an originating router n_o that sent an update or a response with a distance $h_d^{n_o} < h_\infty$ that is the smallest along the path from n_k to d and allowed updates or responses with finite distances to THORP back to n_k before time t_k, such that no router along the path from k to d changes its distance or preferred next hop to d after some time t_k; furthermore, n_o must be a neighbor of destination d. This is a contradiction, because d is not in the connected component of n_k starting at time $t_0 < t_k$, and no router in the connected component can consider itself being a neighbor of d a finite time after t_0. ■

Theorem 6: Reference distances converge to minimum-hop distances at every router within a finite time after network changes stop occurring in a network.

Proof: Assume that all routers in a connected component of a network have converged to their minimum-hop distances for a destination d that can be reached by those routers at time t_o. This implies that $\mathcal{T}$ is satisfied at every router and each router updates its routing state according to Algorithm 1. This implies that $h_d^k = r_d^k$ for every router k.

Assume that all routers in a connected component of a network have converged to a distance of h_∞ to destination d at time t_o. This implies that d is not reachable from those routers. For the sake of contradiction, assume that router k has converged to $r_d^k < h_d^k = h_\infty$ at time $t_k \geq t_o$ and maintains that value of r_d^k indefinitely. Each router $q \in N^k$ must report a distance equal to h_∞ by time t_o, which is a contradiction to $r_d^k < h_d^k = h_\infty$, given that the updates received by k from all its neighbors satisfy $\mathcal{T}$ and hence router k must set $r_d^k = h_d^k$ within a finite time after t_o according to Algorithm 1. ■

VII. Comparing THORP and DUAL against an Optimum Minimum-Hop Routing Method

Performance comparisons of routing protocols are typically done by simulations. Unfortunately, such comparisons render results applicable only to the scenarios used in the simulations, and can be biased depending on such implementation details as the length of timers and topology characteristics. On the other hand, using worst-case performance metrics alone does not provide enough insight on the efficiency of signaling compared to an optimum solution.

To compare DUAL and THORP in a way that is independent of implementation parameters and network topology characteristics, we consider an *Ideal Loop-Free Routing* algorithm (ILR) that somehow updates routing information in the optimum amount of time and incurs the optimum signaling overhead. We define the time optimality (TO) and signaling optimality (SO) of a routing algorithm to be the worst-case additional time in number of steps and worst-case additional signaling in number of messages that the routing algorithm takes to converge compared to ILR.

In the following, w denotes the network diameter in number of hops, A denotes the average degree of a router, and N and E are the number of routers and number of links in the network, respectively. Figure 5 shows a network topology used to discuss worst-case performance of routing algorithms. The numbers next to nodes in the figure denote minimum-hop distances to destination d, solid links with arrowheads are part of shortest paths to d, and links in dashed lines are assumed to exist only when stated.

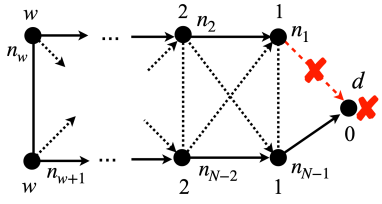

Fig. 5. Example topology for algorithm comparison

The reader should not view the topology of Figure 5 simply as a specific network topology. It is meant as an instrument that takes advantage of the fact that d is the destination to illustrate all cases in which signaling messages must either traverse the diameter of the network or a much shorter path,

depending on the presence of certain links. More complex topologies could be drawn; however, they would all have the same diameter and would include the necessary dashed links that would allow signaling messages not to traverse the entire network diameter.

ILR: We assume that routers executing ILR somehow can differentiate between a neighbor not responding because it failed or because the link to it failed. This ability is not available in a practical distributed routing algorithm, but serves to derive a performance target.

If ILR is executed, it takes only w steps and $N \times A$ messages for all routers to converge to h_∞ as their distances to destination d after the failure of that destination. This is the case independently of the existence of dashed links,

The number of steps and messages needed in ILR after the failure of link (n_1, d) depend on the existence of dashed links. If none exists, it takes $2w$ steps and $2w \times A$ messages for routers n_w to n_1 to correct their routes to d. On the other hand, if link (n_1, n_{N-1}) exists, router n_1 may just select n_{N-1} as next hop and send an update that causes routers to take w steps and $w \times A$ messages for routers n_1 to n_w to correct their routes to d.

DUAL: After the failure of destination d, the diffusing computations started by routers n_1 and n_{N-1} after d fails involve routers along paths of length w; therefore, DUAL takes $\Theta(2w)$ steps and $\Theta(2NA)$ messages to converge in this case.

If link (n_1, d) fails and link (n_1, n_{N-1}) exists and no dashed link (n_i, n_{N-i-1}) exists $(1 \leq i \leq w)$, then the diffusing computation by router n_1 involves routers along paths of length w; therefore, given that n_1 can send an update only after receiving all the replies to its query, DUAL takes $O(3w)$ steps and $O(3NA)$ messages to converge in this case. Accordingly, DUAL has $TO = O(2w)$ and $SO = O(2NA)$.

THORP: If destination d fails, routers take $\Theta(w)$ steps and $O(NA)$ messages to set their distances to d equal to h_∞. This is the case because all routers must receive queries from their next hops stating a distance of h_∞ and a requested distance of 1, the routers must forward those queries, and routers that only have neighbors reporting distances of h_∞ become passive silently.

If link (n_1, d) fails and no link in dashed lines exists, routers take $\Theta(2w)$ steps to correct their routing entries, $w - 1$ steps to block routers n_1 to n_w and w steps to provide them with the new minimum-hop distances based on the distance from n_{w+1} to d, and this incurs $O(2wA)$ messages.

If link (n_1, d) fails and link (n_1, n_{N-1}) exists, router n_1 sends query $Q(d, h_\infty, 1)$. This makes router n_{N-1} send a reply because d reports $h_d^d = 0$ and router n_2 propagates the query, and router n_1 sends update $U(d, 2)$ after processing the reply from n_{N-1}. A query and an update traverse the path from n_1 to n_w with a query stating h_∞ or a finite distance depending on whether a dashed link (n_i, n_{N-i-1}) exists, where $1 \leq i \leq w$. All routers along the path $n_1 \rightarrow n_2 \rightarrow ... \rightarrow n_w$ propagate a query followed immediately by an update, which takes $w + 1$ steps and $O((w + 1)A)$ messages.

Hence, THORP has $TO = \Theta(1)$ and $SO = \Theta(1)$. This means that THORP converges faster than DUAL in most cases. This also indicates that the convergence time of THORP is comparable to or shorter than the convergence time of other minimum-hop distributed routing algorithms.

VIII. Conclusions

THORP is a simple minimum-hop distributed routing algorithm that attains loop-free multi-path routing using only distance information. THORP uses updates, queries and replies like DUAL does. However, it is far more efficient because a query can be resolved with the first response that satisfies the requested distance stated in the query, rather than requiring all neighbors to send their responses. THORP was proven to be correct, and THORP was also shown to be near optimal in terms of its convergence speed.

Our future work focuses on extending the approach introduced in THORP beyond minimum-hop routing, and implementing and testing THORP in specific scenarios (e.g., networks with wired links and wireless networks).

References

[1] B. Albrightson, et al., "EIGRP – A Fast Routing Protocol based on Distance Vectors," *Proc. Interop '94*, 1994.

[2] J. Behrens and J.J. Garcia-Luna-Aceves, "Hierarchical Routing Using Link Vectors," *Proc. IEEE INFOCOM '98*, 1998.

[3] C. Cheng et al., "A Loop-Free Extended Bellman-Ford Routing Protocol without Bouncing Effect," *Proc. ACM SIGCOMM '89*, Aug. 1989.

[4] J. Chroboczek and D. Schinazi, "The Babel Routing Protocol," *RFC 8966*, IETF, Jan. 2021.

[5] T. Clausen et al., "A Critical Evaluation of the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL)," *Proc. IEEE WiMob '11*, Oct. 2011.

[6] J.A. Cobb, "Stabilization of Loop-Free Redundant Routing," *Proc. SSS '07: Ninth Int'l Conf. on Stabilization, Safety, and Security of Distributed Systems*, Nov. 2007.

[7] J.J. Garcia-Luna-Aceves, "Loop-Free Routing Using Diffusing Computations," *IEEE/ACM Trans. Networking*, 1993.

[8] J. J. Garcia-Luna-Aceves and M. Spohn, "Scalable Link-State Internet Routing," *Proc. IEEE ICNP '98*, Oct. 1998.

[9] C.L. Hedrick, "Routing Information Protocol" *RFC 1058*, 1988.

[10] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, Chapter 5, Kluwer Academic Publishers, 1996.

[11] G. Malkin, "RIP Version 2" *RFC 2453*, 1998.

[12] J. Moy, "OSPF Version 2" *RFC 2328*, 1998.

[13] C. E. Perkins and P. Bhagwat, "Routing over Multihop Wireless Network of Mobile Computers," *Proc. ACM SIGCOMM '94*, 1994.

[14] D. Savage et al., "Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)" *RFC 7868*, 2016.

[15] M. Schwartz and T. Stern, "Routing Techniques Used in Computer Communication Networks," *IEEE Trans. Communications*, Apr. 1980.

[16] M. Spohn and J.J. Garcia-Luna-Aceves, "Neighborhood Aware Source Routing," *Proc. ACM MobiHoc '01*, Oct. 2001.

[17] R. Van Glabbeek et al., "Sequence Numbers Do Not Guarantee Loop Freedom–AODV Can Yield Routing Loops," *Proc. ACM MSWiM '13*, Nov. 2013.

[18] S. Vutukury and J.J. Garcia-Luna-Aceves, "A Simple Approximation to Minimum-Delay Routing," *Proc. ACM SIGCOMM '99*, Aug. 31–Sept. 3, 1999.

[19] S. Vutukury and J.J. Garcia-Luna-Aceves, "MDVA: A Distance-Vector Multipath Routing Protocol," *Proc. IEEE INFOCOM '01*, April 2001.

[20] T. Winter, et al., "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," *RFC 6550*, IETF, March 2012.

[21] W.T. Zaumen and J.J. Garcia-Luna-Aceves, "System for Maintaining Multiple Loop-Free Paths between Source Node and Destination Node in Computer Network," U.S. Patent 5,881,243, March 9, 1999.