# SCRIP: Safe and Cycle-Free Multi-Path Routing

J.J. Garcia-Luna-Aceves, Bradley R. Smith, Judith T. Samson
*Computer Science and Engineering Department*
*University of California, Santa Cruz*
Santa Cruz, CA, USA
{jj, brad, jtsamson}@soe.ucsc.edu

***Abstract*—The Safe Cycle-Free Routing Information Protocol (SCRIP) is introduced for routing over multiple acyclic paths based on distance vouchers that attest to the acyclic nature of paths. Routers search and find new shortest paths to destinations without ever creating cycles by trusting updates originated by routers that vouch being closer to destinations. SCRIP is proven to be acyclic and to converge to shortest paths within a finite time even when the network partitions. SCRIP is shown to converge faster than all current shortest-path routing approaches based on destination sequence numbers, inter-nodal coordination, or complete topology information.**

## I. Introduction

The techniques used in modern routing protocols today to guarantee convergence are quite old and consist of using destination sequence numbers [3], establishing multi-hop router coordination [1], or communicating complete topology or complete path information [2]. The design of a routing protocol that never incurs routing-table loops and converges quickly to shortest paths has eluded the research community, and solving this problem has large implications on the performance of most computer networks and the Internet.

The key contribution of this paper is to present the *Safe Cycle-Free Routing Information Protocol* (**SCRIP**), which is the first shortest-path routing protocol that provides multiple routes to destinations and is provably acyclic without incurring long delays converging to optimal routes. Section II describes SCRIP, which ensures that a router sending an update with a valid distance voucher has an acyclic path.

Section III shows that SCRIP is acyclic and converges to optimal routes within a finite time. Section IV shows that SCRIP converges to valid routes faster and more efficiently than any prior shortest-path routing method.

## II. SCRIP

### A. Notation

$N$ is the set of network nodes (routers and destinations), and $E$ is 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 paths to destinations. The $n$th path from 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., $P_d^k(n) = (k, s_d^k(n))P_d^{s_d^k(n)}(m)$. The distance associated with the $n$th path $P_d^k(n)$ from router $k$ to destination $d$ is denoted by $\delta_d^k(n)$.

The distance assigned to destination $d$ is denoted by $\delta_0 = 0$, and the distance assumed for any unreachable destination is denoted by $\delta_\infty$.

The variable $r_d^k$ denotes the *reference distance* for destination $d$ at router $k$.

The value of the distance voucher for destination $d$ at router $k$ is denoted by $\nu_d^k$. The value of a distance voucher is set to true (T) if the voucher attests that the associated path is acyclic, and is set to false (F) otherwise.

The best next hop to destination $d$ at router $k$ is denoted by $s_d^k$, with $s_d^k = 0$ denoting the case that there is no next hop to $d$, and $S_d^k$ is the set of neighbors of router $k$ that can be used as next hops to destination $d$.

### B. Information Exchanged

Routers exchange routing messages reliably among one another 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 updates. An **update** from router $k$ is a tuple denoted by $U(d, n_d^k, \nu_d^k, \rho_d^k, \delta_d^k)$, where $n_d^k$ is the Identifier of the main recipient of the update, and $\rho_d^k$ s a reference distance that states the distance of an attested acyclic path if $\nu_d^k = T$ or the maximum requested distance of an acyclic path being requested if $\nu_d^k = F$.

The following conventions are used to interpret the information carried in an update:

- $n_d^k = 0$ denotes all neighbors of router $k$.
- $n_d^k = k$ with $d = k$ indicates that the update is a "hello" to refresh the presence of node $k$.
- $n_d^k = q \neq k$ indicates that $q$ is the primary recipient of the update; however, all routers receiving the update process it.

### C. Information Maintained

Each router $k$ knows its own identifier ($k$) and maintains a Link-Weight Table ($LWT^k$), a Neighbor Table ($NT^k$), a Routing Table ($RT^k$), and its initialization status ($\sigma^k$).

$LWT^k$ lists an entry for each link to a known neighbor router $n \in N^k$. The entry for link $(k, n)$ in $LWT^k$ states: (a) The weight $l(k, n)$ of the link, and (b) 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.

$NT^k$ lists the distance and distance voucher reported by each neighbor for each destination of interest. The entry in $NT^k$ for destination $d$ at router $k$ is denoted by $NT^k(d)$ and

978-1-6654-8534-0/22/$31.00 ©2022 IEEE

specifies for each neighbor $p \in N^k$ the latest distance $\delta_{dp}^k$ and value of the distance voucher $\nu_{dp}^k$ for the neighbor.

If a neighbor $q$ has not reported any distance for $d$ to router $k$, then router $k$ assumes that $\delta_{dq}^k = \delta_\infty$ and $\nu_{dq}^k = T$.

$RT^k$ lists an entry for each destination $d$ for which the router must maintain routing information. The entry in $RT^k$ for destination $d$ is denoted by $RT^k(d)$ and states: $r_d^k$, $\delta_d^k$, $\nu_d^k$, $s_d^k$, $S_d^k$, ND $[p_d^k, \lambda_d^k]$. The latter is a search tuple where $p_d^k$ denotes the router or routers that sent a search request for $d$, and $\lambda_d^k$ is the smallest reference distance stated in an ongoing search regarding $d$.

The search tuple $[p_d^k, \lambda_d^k]$ is interpreted as follows:
• $p_d^k = 0$ and $\lambda_d^k = \delta_\infty$ states that router $k$ is not part of an active search regarding $d$.
• $p_d^k = q \neq k$ and $\lambda_d^k < \delta_\infty$ states that $k$ forwarded a search on behalf of router $q$ and $\lambda_d^k$ is the reference distance used in the search.
• $p_d^k = 0$ and $\lambda_d^k < \delta_\infty$ states that $k$ has been part of multiple searches regarding $d$ and $\lambda_d^k$ is the smallest reference distance used in such searches.

$\sigma^k$ is true (T) if router $k$ has been properly initialized, which means that an initialization period longer than $LT$ (e.g., $5LT$) has elapsed since the router started or rebooted, and is false (F) otherwise.

### D. SCRIP Operation

This description assumes that routers send routing messages reliably after waiting for short or long time intervals, and that all destinations are of interest to all routers. Slightly different mechanisms would be needed if unreliable message transmissions were used and not all destinations were of interest to all routers.

*1) Initialization:* A router $k$ is initialized with a defined neighbor set, which is applicable for a computer network with wired links. The routing state for the router is started with $\sigma^k = T$, $\delta_k^k = r_k^k = \delta_0$, and $s_k^k = k$ for $k$ itself, and for each defined neighbor $q \in N^k$: $\delta_{kq}^k = \delta_\infty$, $\nu_{kq}^k = T$, $\delta_{qq}^k = \delta_\infty$, and $\nu_{qq}^k = T$, to indicate that no update has been received from $q$. Router $k$ immediately sends a routing message with a single update $U(k, 0, T, \delta_0, \delta_0)$.

*2) 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 needed to reflect changes to its routing table, a router sends a message with a "hello" update equals $U(k, k, T, \delta_0, \delta_0)$ to simply updates 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.

*3) Link Changes:* Router $k$ updates $LWT^k$ when an adjacent outgoing link $(k, q)$ changes its weight $l(k, q)$, and updates $NT^k$ and $RT^k$ 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 LWT entry expires, or a routing message $M^q$ is received from a neighbor.

A router $k$ detects that a neighbor $q$ just became active when it receives a "hello" update from $q$ and its local state for $q$ has $\delta_{qq}^k = \delta_{kq}^k = \delta_\infty$, which indicates that no messages were being received over link $(k, q)$. In this case, router $k$ immediately sends a routing message for each destination for which it can offer a valid voucher.

*4) Voucher State:* Router $k$ can be in one of two routing states for a destination $d$ depending on the value of its voucher $\nu_d^k$. Router $k$ has a valid voucher for $d$ ($\nu_d^k = T$) only if it can either: (a) vouch for one or more acyclic paths to destination $d$ because one or more of its neighbors has reported a valid voucher and a distance that is smaller than its own reference distance ($r_d^k$); or (b) state that it cannot reach $d$ because all its neighbors have reported a distance equal to $\delta_\infty$. This condition can be stated formally as follows:

$$\mathcal{V} : \Big(\exists\ q \in N^k([r_d^k > \delta_{dq}^k] \wedge [\nu_{dq}^k = T])) \vee (\forall q \in N^k(\delta_{dq}^k = \delta_\infty)\Big) \tag{1}$$

If $\mathcal{V}$ is true then router $k$ updates its routing state as follows:

$$\nu_d^k \leftarrow T;\ S_d^k \leftarrow \{q \in N^k \mid (r_d^k > \delta_{dq}^k) \wedge (\nu_{dq}^k = T)\}; \tag{2}$$

$$\delta_d^k \leftarrow Min\{\delta_{dn}^k + l(k, n) \mid n \in S_d^k\};$$

$$s_d^k \leftarrow Min\{n \in S_d^k \mid \delta_{dn}^k + l(k, n) = \delta_d^k\};$$

**if** $([\nu_d^k = T] \wedge [\ \forall q \in N^k([\nu_{dq}^k = T] \vee [\delta_{dq}^k = \delta_\infty])])$ **then** $r_d^k \leftarrow \delta_d^k$

It follows from Eq. (2) that the value of $r_d^k$ can be set equal to $\delta_d^k$ only when its own voucher is valid and either all its neighbors provide valid vouchers or all its neighbors declared $d$ to be unreachable. Router $k$ may use any neighbor in $S_d^k$ as a next hop to destination $d$, and keeps a preferred next-hop neighbor $s_d^k$.

Router $k$ cannot vouch for acyclic paths to destination $d$ if $\mathcal{V}$ is false. In this case, router $k$ must set $\nu_d^k = F$ and start a search for a neighbor router or a remote router that can vouch for an acyclic path that is shorter than the distance value at $k$ when $\mathcal{V}$ was still satisfied, which equals $r_d^k$. Router $k$ tracks which neighbor can vouch for an acyclic path that router $k$ can use to make $\mathcal{V}$ true again, and resets all the vouchers from its neighbors to $F$ to start its search. Furthermore, to avoid routing cycles, router $k$ does not change its best next hop to $d$ (if it has any) until it obtains at least one response to its search that makes $\mathcal{V}$ true again. This can be stated as follows:

$$\nu_d^k \leftarrow F;\ \forall q \in N^k(\nu_{dq}^k \leftarrow F); \tag{3}$$

$$\delta_d^k \leftarrow \delta_{ds_d^k}^k + l(k, s_d^k);\ \ \textbf{if}\ (\delta_d^k = \delta_\infty)\ \ \textbf{then}\ \ s_d^k \leftarrow 0$$

Once $\mathcal{V}$ is false, an update from any neighbor $q$ with a valid voucher ($\nu_d^q = T$) and a reference distance that is shorter than

the reference distance at $k$ ($r_d^k > \rho_d^q$) makes $\mathcal{V}$ true again, because the path from $q$ to $d$ consists of the concatenation of an acyclic path from $q$ to another router $p$ that attests to the acyclic nature of a path to $d$ through a next hop $n$ such that $\delta_{dn}^p = \rho_d^q < r_d^k$. Alternatively, to account for $d$ being unreachable, router $k$ sets $\mathcal{V}$ to true again if all its neighbors declare $d$ to be unreachable (see Eq. (1)).

*5) Interpreting Vouchers from Neighbors:* Router $k$ interprets a valid voucher from a neighbor $q$ as an independent attestation to the acyclic nature of a path from $q$ to $d$ if $\mathcal{V}$ is true. However, router $k$ resets all its neighbor vouchers to $F$ when $\mathcal{V}$ becomes false. Accordingly, while $\mathcal{V}$ is false router $k$ must interpret a valid voucher from a neighbor $q$ as a relative attestation whose truth value at $k$ depends on whether the path offered by $q$ takes into account the reference distance $r_d^k$ communicated by $k$ in its request. This is stated formally as follows as part of the method used to update $NT^k$:

$$\delta_{dq}^k \leftarrow \delta_d^q; \quad \textbf{if } (\nu_d^k = T) \textbf{ then } \nu_{dq}^k \leftarrow \nu_d^q; \tag{4}$$

$$\textbf{if } ([\nu_d^k = F] \wedge [\nu_d^q = T] \wedge [r_d^k \geq \rho_d^q]) \textbf{ then } \nu_{dq}^k \leftarrow T$$

*6) Sending Updates and Search Requests:* Routers use updates with valid distance vouchers to inform their neighbors of new distance values, and use updates with invalid distance vouchers to start or propagate search requests for acyclic paths to destinations.

The steps that router $k$ takes in response to an input event depend on whether $\mathcal{V}$ in Eq. (1) is true *after* $NT^k$ has been updated according to Eq. (4) as a result of the input event.

**$\mathcal{V}$ is true after input event**: In this case, router $k$ updates its routing state according to Eq. (2) and sends an update, sends a response, or forwards a request depending on the value of $\delta_d^k$, its search tuple $[p_d^k, \lambda_d^k]$, and the update that may be received from a neighbor $q \in N^k$.

**(i)** *Router $k$ detects a link-cost change or receives an update $U(d, n_d^q, T, \rho_d^q, \delta_d^q)$ from $q$*:

If $\delta_d^k < \delta_\infty$ before the input event, then router $k$ sends an update if the value of $\delta_d^k$ changes or $k$ needs to send a response to a prior request. Router $k$ knows that it must send a response to a prior request if: (a) $\lambda_d^k < \delta_\infty$, which indicates that router $k$ has a pending request; and (b) $\rho_d^q < \lambda_d^k$, which indicates that a remote router vouches for an acyclic path that satisfies the pending request. In this case, its update $U(d, n_d^k, T, \rho_d^k, \delta_d^k)$ states $n_d^k = p_d^k$ and $\rho_d^k = \rho_d^q$; router $k$ also resets its search tuple $[p_d^k, \lambda_d^k]$ to $[0, \delta_\infty]$. If $\delta_d^k$ changed its value and router $k$ is neither part of a search ($\lambda_d^k = \delta_\infty$) nor can it satisfy a pending request ($\rho_d^q \geq \lambda_d^k$) , then router $k$ sends an update $U(d, 0, T, r_d^k, \delta_d^k)$.

If $\delta_d^k = \delta_\infty$ before the input event, then router $k$ updates its routing state according to Eq. (2) and sends an update $U(d, n_d^k, T, \rho_d^k, \delta_d^k)$ with $n_d^k = p_d^k$ and $\rho_d^k = r_d^k$ if either the value of $\delta_d^k$ changed ($\delta_d^k < \delta_\infty$) or the input event allowed $\mathcal{V}$ to be satisfied, which router $k$ identifies because $r_d^k < \delta_\infty$ before the input event in that case.

**(ii)** *Router $k$ receives a request $U(d, n_d^q, F, \rho_d^q, \delta_d^q)$ from $q$*:

Router $k$ can vouch for an acyclic path that is shorter than the reference distance in the request if the following condition is satisfied:

$$\mathcal{R} : (\nu_d^k = T) \wedge (\rho_d^q > \delta_{ds_d^k}^k) \tag{5}$$

If $\mathcal{R}$ in Eq. (5) is satisfied, then router $k$ sends an update $U(d, q, T, \rho_d^k, \delta_d^k)$ with $\rho_d^k = \delta_{ds_d^k}^k$ as a response to $q$.

If $\mathcal{R}$ in Eq. (5) is not satisfied, then router $k$ propagates or stops the search depending on the values of $[p_d^k, \lambda_d^k]$ and $n_d^q$, and sends an update depending on the value of $\delta_d^k$.

If router $k$ was asked to help (i.e., $n_d^q = k$ or $n_d^q = 0$) and $\rho_d^q < \lambda_d^k$, then router $k$ forwards request $U(d, s_d^k, F, \rho_d^k, \delta_d^k)$ with $\rho_d^k = \rho_d^q$, sets $p_d^k = q$ if $\lambda_d^k = \delta_\infty$ to remember that a response is needed for $q$, and sets $\lambda_d^k = Min\{\lambda_d^k, \rho_d^q\}$ to remember the smallest reference distance stated in a request.

If router $k$ was asked to help and $\rho_d^q \geq \lambda_d^k$, then router $k$ stops the request because it already has forwarded a prior request with a reference distance that satisfies the current request from $q$. In this case, $k$ sets $p_d^k = 0$ to ensure that a future response is sent back to all its neighbors. Furthermore, router $k$ sends update $U(d, q, T, \rho_d^k, \delta_\infty)$ in which $\rho_d^k = \rho_d^q$ if $\delta_d^k = \delta_\infty$ to inform $q$ that it cannot reach $d$, and sends update $U(d, 0, T, r_d^k, \delta_d^k)$ if $\delta_d^k$ changed in value and $\delta_d^k < \delta_\infty$ to inform all its neighbors of its new shortest distance to $d$.

If router $k$ was not asked to help (i.e., $n_d^q \neq k$ and $n_d^q \neq 0$) and $\mathcal{R}$ is not satisfied, router $k$ sends an update in two cases only. Router $k$ sends update $U(d, 0, T, r_d^k, \delta_d^k)$ if $\delta_d^k$ changed as a result of the request from $q$ and $\delta_d^k < \delta_\infty$. Alternatively, router $k$ sends response $U(d, q, T, \rho_d^k, \delta_d^k)$ with $\rho_d^k = \rho_d^q$ if $\delta_d^k = \delta_\infty$ to inform $q$ that it cannot reach $d$.

**$\mathcal{V}$ is false after input event**: In this case, router $k$ updates $LWT^k$ and $NT^k$ as needed, updates $RT^k$ as stated in Eq. (3), and sends an update, a response, or forwards a request depending on the value of $\delta_d^k$, its search tuple $[p_d^k, \lambda_d^k]$, and the update that may be received from a neighbor $q$.

**(i)** *Router $k$ detects a link-cost change or receives an update $U(d, n_d^q, T, \rho_d^q, \delta_d^q)$ from $q \in N^k$*:

Router $k$ sends a request $U(d, n_d^k, F, \rho_d^k, \delta_d^k)$ with $\rho_d^k = r_d^k$ and $n_d^k = s_d^k$ if $r_d^k < \lambda_d^k$, because any pending request does not satisfy its own reference distance. If this is the case, router $k$ updates $\lambda_d^k = Min\{\lambda_d^k, r_d^k\}$ to remember the smallest reference distance used in a request, and updates $p_d^k = k$ if $\lambda_d^k = \delta_\infty$ or $p_d^k = 0$ if $\lambda_d^k < \delta_\infty$ to remember how to forward responses to its request. On the other hand, if $r_d^k \geq \lambda_d^k$, router $k$ knows that it has sent a prior request with a smaller reference distance than its own and simply updates $p_d^k = 0$ to remember that it is involved in multiple requests.

**(ii)** *Router $k$ receives a request $U(d, n_d^q, F, \rho_d^q, \delta_d^q)$ from $q$*:

If $\rho_d^q < \lambda_d^k$, then router $k$ sends a request $U(d, 0, F, \rho_d^k, \delta_d^k)$ to all its neighbors with $\rho_d^k = \rho_d^q$, and updates $\lambda_d^k = \rho_d^q$ and $p_d^k = 0$. Otherwise, if $\rho_d^q \geq \lambda_d^k$, router $k$ knows it has sent a request that satisfies the request from $q$. In this case, router $k$ updates $p_d^k = 0$ to remember that it is involved in multiple requests, and sends update $U(d, q, T, \rho_d^k, \delta_d^k)$ with $\rho_d^k = \rho_d^q$ if $\delta_d^k = \delta_\infty$ to inform $q$ that it cannot reach $d$.

### E. Example of SCRIP Operation

Figure 1 illustrates the fast convergence of SCRIP in a five-node network. The reference distance, distance, and next hop to destination $d$ are indicated next to each router. In contrast to the other methods, routers may have multiple next hops to destinations, as indicated by arrowheads. An update sent by router $k$ omits destination $d$ and is denoted by $U[n_d^k, \nu_d^k, \rho_d^k, \delta_d^k]$ and the Boolean values of true and false for vouchers are denoted by T and F, respectively.

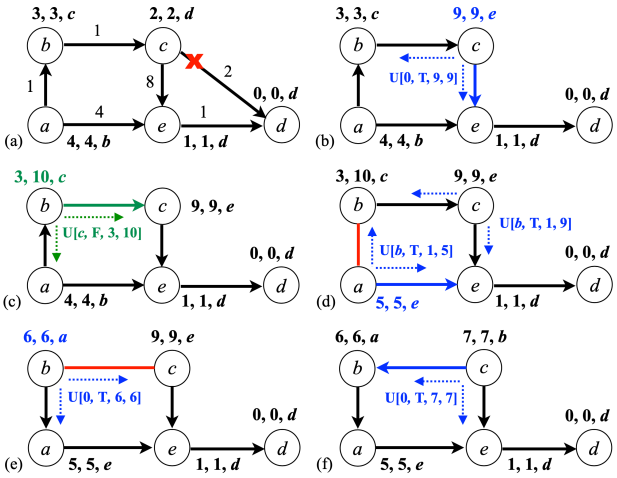


Figure 1: Fast convergence without blocking in SCRIP

Figure 1(b) shows that router $c$ sends an update with a valid distance voucher after the failure of link $(c, d)$ because its distance increases to 9 but its neighbor $e$ satisfies $\mathcal{V}$ in Eq. (1) with $2 = r_d^c > \delta_{de}^c = 1$ and $\nu_{de}^c = T$.

As Figure 1(c) shows, the update from router $c$ makes $\mathcal{V}$ not be satisfied by any neighbor of $c$. This causes router $b$ to start a search for a router that can vouch for an acyclic path. Accordingly, the update from $b$ is a request for a router to vouch for a path shorter than $r_d^b = 3$ and states $\nu_d^b = F$, $\rho_d^k = 3$, and $\delta_d^b = 10$. Router $b$ keeps $s_d^b = c$ while its search proceeds. This is the case because $\delta_{dc}^b + l(b, c) = 10 < \delta_\infty$; however, router $b$ cannot change its next hop until it receives an update vouching for an acyclic path shorter than $r_d^b = 3$.

Figure 1(d) shows that routers $c$ and $a$ send back updates to $b$ with valid distance vouchers. This is the case because both routers have neighbors that reported a distance smaller than the reference distance $\rho_d^b = 3$ in the request from $b$ and hence $\mathcal{V}$ is satisfied at both routers.

Figures 1(e) shows that the updates from $a$ and $c$ allow router $b$ to set $\delta_d^b = 6$ and $\nu_d^b = T$. Accordingly, router $b$ sets $r_d^b = \delta_d^b = 6$ because all its neighbors provided updates with valid vouchers and reference distances smaller than $r_d^b = 3$. Figures 1(f) shows that router $c$ decreases its distance and reference distance to 7 after it processes the update from $b$.

This example illustrates that SCRIP is acyclic while routers converge quickly to optimal paths. A router that needs to search for routers that can vouch for acyclic paths can trust the first update with a valid distance voucher and a reference distance that is shorter than its own reference distance.

## III. SCRIP Correctness

The following theorems make use of the following nomenclature and formalize the previous argument to prove that SCRIP is acyclic and converges to optimal paths within a finite time.

$\mathcal{U}$ denotes the proposition that router $k$ executes SCRIP according to the specification provided in Section IV while having a valid voucher for destination $d$.

$\mathcal{W}$ denotes the proposition that router $k$ executes SCRIP according to the specification provided in Section IV while waiting for a valid voucher for destination $d$.

$\mathcal{C}$ denotes the proposition that SCRIP is executed at every router for destination $d$. according to the specification provided in Section IV, i.e., $\mathcal{C} \equiv \mathcal{U} \vee \mathcal{W}$.

$\mathcal{A}$ denotes the proposition that a routing protocol is acyclic.

$\mathcal{O}$ denotes the proposition that a routing protocol converges to paths with optimum weights, and it is true that $\delta_d^k = \omega_d^k$ for any destination $d$ at each router $k$.

$\mathcal{P}$ denotes the proposition that valid distance vouchers propagate through acyclic paths.

$\mathcal{I}$ denotes the proposition that all routers in a connected network component converge to $\delta_\infty$ for a destination $d$ that is not reachable from the network component.

$\mathcal{E}$ denotes the proposition that the reference distance equals the shortest distance at each router for a destination $d$ within a finite time after network changes stop occurring in a network.

*Theorem 1:* A path in which $\mathcal{U}$ is true at every router cannot be a loop.

*Proof:* Assume that $\mathcal{U}$ 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 $\mathcal{V}$ 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 $s_d^{v_i}(t) = s_d^{v_i}(t_i^+)$, $\delta_d^{v_i}(t) = \delta_d^{v_i}(t_i^+)$, 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 $\mathcal{U}$ is true at each router $v_i \in L$: (a) $r_d^{v_i}(t_i^+) = r_d^{v_i}(t) > \delta_{dv_{i+1}}^{v_i}(t)$; (b) $\delta_{dv_{i+1}}^{v_i}(t) = \delta_{dv_{i+1}}^{v_i}(t_i^+) = \delta_d^{v_{i+1}}(t_{i+1})$; and (c) $\delta_d^{v_{i+1}}(t_{i+1}) = r_d^{v_{i+1}}(t_{i+1}) = r_d^{v_{i+1}}(t)$. It follows from (a), (b) and (c) that $r_d^{v_i}(t) > r_d^{v_{i+1}}(t)$. However, these results constitute a contradiction, because they imply that $r_d^{v_i}(t) > r_d^{v_i}(t)$ for all $v_i \in L$; therefore, the theorem is true. ■

*Theorem 2:* No routing-table loop can be created by a router for which $\mathcal{W}$ is true.

*Proof:* A router that determines that $\mathcal{V}$ is not satisfied and sends a request for valid vouchers 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 for which $\mathcal{U}$ is true, which negates the existence of a routing-table loop because of Theorem 1. ■

*Theorem 3:* SCRIP is acyclic for any destination $d$.

*Proof:* If $\mathcal{U}$ is true at every router, then it follows from Theorem 1 that no routing-table loops can form. It also follows from Theorem 2 that no router for which $\mathcal{W}$ is true can create a loop. Therefore, the proof needs to show that a router for which $\mathcal{W}$ is true and then receives valid voucher and selects a next hop to destination $d$ can create a loop.

Once $\mathcal{W}$ is true for a given router $k$, the router must receive an update with a valid voucher, and a router $n \in N^k$ can send a response to router $k$ only if it has a valid voucher itself. The path from $n$ to $d$ either consists only of routers for which $\mathcal{U}$ is true, or consists of routers for which either $\mathcal{U}$ or $\mathcal{W}$ is true. 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 for which $\mathcal{U}$ is true or $\mathcal{W}$ is true, 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. ■

The following theorems show that SCRIP converges to optimal distances for destinations that can be reached, and to $\delta_\infty$ for unreachable destinations.

*Theorem 4:* Valid distance vouchers propagate through acyclic paths.

*Proof:* The proof needs to show that $\mathcal{C} \rightarrow \mathcal{P}$. By definition, a valid distance voucher attests that a path is acyclic and hence $\mathcal{A} \rightarrow \mathcal{P}$. From Theorem 3, $\mathcal{C} \rightarrow \mathcal{A}$ . Therefore, the theorem is true because $(\mathcal{C} \rightarrow \mathcal{A}) \wedge (\mathcal{A} \rightarrow \mathcal{P}) \rightarrow (\mathcal{C} \rightarrow \mathcal{P})$ ■

*Theorem 5:* SCRIP converges to optimal routes for any given destination within a finite time after network changes stop occurring in a finite connected network.

*Proof:* The proof must show that $\mathcal{C} \rightarrow \mathcal{O}$ for any destination $d$. The proof proceeds by showing that $\mathcal{C} \wedge \neg\mathcal{O}$ is a contradiction. Accordingly, assume that SCRIP works correctly and also assume that router $k$ converges incorrectly with $\delta_d^k > \delta_d^{\star k}$, where $\delta_d^{\star k}$ is the optimum distance from router $k$ to destination $d$.

Given that the network is connected and finite, all acyclic paths in the network are finite and it takes a finite time for updates starting from destination $d$ to propagate over any acyclic path. Distance $\delta_d^{\star k}$ corresponds to an acyclic path from $k$ to $d$ through some neighbor router $s \in N^k$ because it is optimum. From Lemma 1 and the fact that $\mathcal{C}$ is true, this implies that router $k$ must receive an update from router $s$ reporting $\delta_d^{\star k}$ and a valid voucher for destination $d$. However, this is a contradiction to $\mathcal{C}$ being true, because then Eqs. (4) and (5) would require router $k$ to make $\delta_d^{\star k}$ its distance to destination $d$. ■

*Theorem 6:* SCRIP converges to $\delta_\infty$ at every router of a finite connected-network component for any destination that is unreachable from the network component within a finite time after network changes stop occurring.

*Proof:* The proof needs to show that $\mathcal{C} \rightarrow \mathcal{I}$ and proceeds by showing that $\mathcal{C} \wedge \neg\mathcal{I}$ is a contradiction.

Assume that no network changes occur after time $t_n$ and $\neg\mathcal{I}$ is true with at least one router $n_k$ converging to a finite distance for $d$ at time $t_k \geq t_n$. Because $\mathcal{C}$ is true, any router $n_i$ that has a valid voucher and a distance $\delta_d^{n_i} < \delta_\infty$ at time $t_k$ must have a next-hop neighbor $n_{i-1} \in N^{n_i}$ such that $\delta_{dn_{i-1}}^{n_i} < \delta_d^{n_i}$ and $\nu_{dn_{i-1}}^{n_i} = T$. It follows from Theorem 3 that there must be an originating router $n_o$ that issued a valid voucher and a distance $\delta_d^{n_o} < \delta_\infty$ that allowed updates with valid vouchers and finite distances to propagate along one or more acyclic paths to $n_k$ before time $t_k$. Furthermore, because $\mathcal{C}$ is true, router $n_o$ must be a neighbor of destination $d$. However, this is a contradiction because $d$ is not in the connected component starting at least at time $t_0$, and hence no router in the connected component can consider itself being a neighbor of $d$ after some finite time $t_o \geq t_n$. ■

*Theorem 7:* The reference distance used in SCRIP for any given destination converges to the correct distance value for that destination at every router within a finite time after network changes stop occurring in a network.

*Proof:* The proof needs to show that $(\ \mathcal{C} \wedge [\mathcal{O} \vee \mathcal{I}]\ ) \rightarrow \mathcal{E}$, which is equivalent to $(\ [\mathcal{C} \wedge \mathcal{O}] \rightarrow \mathcal{E}) \wedge ([\mathcal{C} \wedge \mathcal{I}] \rightarrow \mathcal{E})$.

$(\mathcal{C} \wedge \mathcal{O}) \rightarrow \mathcal{E}$: Assume that all routers have converged to their optimum distances for any given destination $d$ at time $t_o$. This implies that $\mathcal{V}$ is satisfied at every router and each router updates its routing state according to Eq. (2). Router $k$ can have $r_d^k < \delta_d^k$ indefinitely after time $t_o$ only if at least one neighbor reports an invalid voucher for $d$ indefinitely, which is a contradiction to the correct operation of SCRIP and the fact that $\mathcal{V}$ is satisfied at every router.

$(\mathcal{C} \wedge \mathcal{I}) \rightarrow \mathcal{E}$: Assume that all routers have converged to a distance of $\delta_\infty$ to destination $d$ at time $t_o$. This implies that $d$ is not reachable from any router because $\mathcal{C}$ is true. For the sake of contradiction, assume that router $k$ has converged to $r_d^k < \delta_d^k = \delta_\infty$ at time $t_k \geq t_o$ and maintains that value of $r_d^k$ indefinitely. Because $\mathcal{C}$ is true, each router $q \in N^k$ must report a distance equal to $\delta_\infty$ by time $t_o$, which is a contradiction to $r_d^k < \delta_d^k = \delta_\infty$, given that the updates received by $k$ from all its neighbors satisfy $\mathcal{V}$ and hence router $k$ must set $r_d^k = \delta_d^k$ according to Eq. (2) within a finite time after $t_o$. ■

## IV. Performance Comparison

SCRIP is compared with two existing methods for acyclic routing and the topology-broadcast method, which is not acyclic but is known to have fast convergence after distance increases.

The time complexity $TC$ (worst-case number of steps) and communication complexity $CC$ (worst-case number of messages) needed by all routers to obtain acyclic paths to a given destination after a single link-weight increase are strong indicators of the signaling overhead of a routing protocol.

The blocking/looping complexity $BC$ is the worst-case time during which blocking or looping occurs after a link-weight increase before all routers have valid routing state. It is a strong indicator of the speed with which a routing protocol restores valid routing state after input events.

$TC$, $CC$ and $BC$ are used to compare the performance of SCRIP and other shortest-path routing methods independently of implementation or topology parameters that may bias comparisons by simulation. In the following, $H$ denotes the network diameter in number of hops, $A$ denotes the average degree of a router, $T$ denotes a timer delay that by design must be much longer than the time an update takes to traverse the network diameter, and $N$ and $E$ are the number of routers and number of links in the network, respectively.

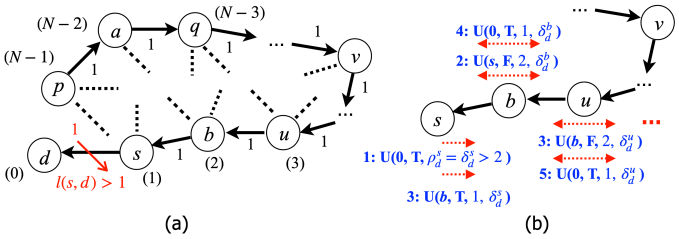


Figure 2: Example of worst-case link-weight increase

Fig. 2 shows an example of a single link-weight increase that impacts distances to destination $d$ at all other routers. The numbers in parenthesis in Fig. 2(a) denote distances to $d$, solid links with arrowheads are part of shortest paths to $d$ and each has a weight of 1, and links in dashed lines are assumed to have large weights that make the chain involving all nodes other than $d$ correspond to the shortest paths to $d$.

**Inter-Nodal Coordination** ($c$): Using diffusing computations is more efficient than other techniques based on inter-nodal coordination, and is the technique assumed in this analysis. A single link-weight increase may cause the start of a diffusing computation that causes queries, replies, and final updates to traverse the directed acyclic graph (DAG) formed by the routers affected by the computation. In the worst case (Fig. 3), the DAG involves most network links and $N-1$ routers (see Fig. 3). Therefore, $TC_c = O(3N)$ and $CC_c = O(6E)$. These are loose bounds, because convergence may occur after an update traversing a single hop [1].

$BC_c = 0$ after a link-weight increase in DUAL, because routers continue using their current next hops while they wait for replies. However, the source of a diffusing computation caused by a link or node failure remains blocked until all its neighbors send their replies (see Fig. 2). Accordingly, $BC_c = O(2N)$ in this case, because a query and its replies must traverse a DAG whose length may be $O(N)$.

**Topology Broadcasting** ($t$): In this method, a single link-weight change requires link-state updates (LSU) to be flooded in the network over all links, independently of next-hop routing choices. Therefore, $CC_t = \Theta(E)$. However, routers may have incorrect routing-table entries until a LSU with the most recent sequence number is stored at every router, and the change may occur such that almost an entire timer delay $T$ is incurred waiting for the periodic transmission of the next sequence number from the head of the link issuing the LSU. If all links have similar delays, then $TC_t = O(H+T)$, because the paths traversed by LSU's have $O(H)$ length. If link weights reflect link delays, the paths traversed by LSU's may have $O(N)$ length and hence $TC_t = O(N+T)$.

As Fig. 3 illustrates, routing loops may occur until all routers store the same data in their topology tables. The steps incurred for an LSU to reach all other routers is $O(H+T)$ if all links have similar delays, and $O(N+T)$ if link weights reflect their delays; therefore, $BC_t = O(N+T)$.

**Distance Vouchers** ($v$): When the weight of link $(s,d)$ increases in Fig. 3, router $s$ sends update to all its neighbors stating a larger distance. This causes router $b$ to send request $U[s, F, \rho_d^b, \delta_d^b]$ with $\rho_d^b = r_d^b = 2$ to $s$ to search for a valid acyclic path (see Fig 3(b)). Router $s$ responds with update $U[b, T, 0, \delta_d^s]$ because $\delta_{dd}^s = 0 < 2$, and router $u$ sends a request $U[b, F, \rho_d^u, \delta_d^u]$ to $b$ with $\rho_d^u = \lambda_d^u = 2 = Min\{2, \delta_\infty\}$. The response from $s$ allows $b$ to send update $U[0, T, 0, \delta_d^b]$ to $u$ and all its neighbors reporting a valid path, and this in turn allows $u$ to send $U[0, T, 0, \delta_d^u]$. The same request-update process takes place at each hop along the reverse path $P_{ps}$, which has $N-2$ hops, taking a total of $N+1$ steps. Therefore, $TC_v = O(N)$ and $CC_v = O(E)$, given that updates are sent to all neighbors. These are loose bounds, because a search after a weight increase may involve a single hop (e.g., the weight of link $(s,d)$ increases in Fig. 3) or very few hops.

$BC_c = 0$ after a link-weight increase in SCRIP, because in this case routers continue using their current next hops while they wait for responses with valid vouchers. A link or node failure may prevent a router from using any neighbor as a next hop and force it to search for remote routers with valid vouchers. The resulting request-update process may take place over paths of length $O(N)$; therefore, $BC_c = O(N)$ in this case. However, this is a loose bound, because a search may reach routers with valid vouchers in just $h$ hops, with $h \ll N$, and take only $O(h)$ steps.

These results show that SCRIP incurs less signaling overhead and converges to acyclic paths faster than all prior shortest-path routing methods.

## V. Conclusions

The Safe Cycle-Free Routing Information Protocol (SCRIP) was introduced. Its novelty consists of: (a) Replacing destination sequence numbers with distance vouchers and reference distances that allow routers to attest that they are closer to destinations than those routers seeking to establish new acyclic paths; and (b) using inter-nodal coordination *after* routing tables are updated with valid routes to allow routers to increase their reference distances. SCRIP was proven to be correct and loop-free, and it was shown to be more efficient and to converge faster to shortest paths than routing protocols based on destination sequence numbers, inter-nodal coordination, or topology broadcasting.

## References

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

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

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