Stepwise Assertional Design of Distance–Vector Routing Algorithms*

Cengiz Alaettinoğlu, A. Udaya Shankar

Institute for Advanced Computer Studies
and Department of Computer Science
University of Maryland
College Park, Maryland 20742
ca@isi.edu, shankar@cs.umd.edu

UMIACS-TR-92-39, CS-TR-2869
August 1993

Abstract

There are many kinds of distance–vector algorithms for adaptive routing in wide-area computer networks, ranging from the classical Distributed Bellman–Ford to several recent algorithms that have better performance. However, these algorithms have very complicated behaviors and their analyses in the literature has been incomplete (and operational). In this paper, we present a stepwise assertional design of a recently proposed distance–vector algorithm. Our design starts with the Distributed Bellman–Ford and goes through two intermediate algorithms. The properties established for each algorithm hold for the succeeding algorithms. The algorithms analyzed here are representative of various internetwork routing protocols.

* This work is supported in part by National Science Foundation Grant No. NCR 89-04500, and by RADC and DARPA under contract F30602-90-C-0010 to UMIACS at the University of Maryland. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, RADC, or the U.S. Government.
## Contents

1 Introduction  

2 Preliminaries: System Model and Proof Rules  

3 Algorithm A1  

4 Algorithm A2  

5 Algorithm A3  

6 Algorithm A4  

7 Concluding Remarks
1 Introduction

Adaptive routing protocols are responsible for choosing optimal routes for data packets in wide-area store-and-forward computer networks such as the Internet. In these networks, each link has a cost (indicating the current traffic on the link) that changes with time; furthermore, links can fail and recover. We refer to such changes as **topology changes**. A routing protocol must monitor these topology changes and adapt its routes accordingly.

In a routing protocol, each node maintains for each destination a neighboring node id, referred to as its **next-hop**. The node forwards data packets destined for the destination to its next-hop. The next-hop can be *nil*, in which case the node does not know where to forward data packets for that destination. The objective can be informally stated as follows: (a) the succession of next-hops for the destination from any node should lead to the destination (unless the destination is unreachable); and (b) the cost of this next-hop path should be minimum amongst all paths from the node to the destination.

A popular approach to routing is the **distance-vector** approach, which is based on the Bellman-Ford algorithm [5]. In this approach, each node maintains for each destination a set of **distances**, one for each of its neighbors, and chooses a neighbor with minimum distance as the next-hop. Thus, each node requires $O(N \times \epsilon)$ space, where $N$ is the number of nodes in the network and $\epsilon$ is the average degree of a node. However it is well known that the straight-forward distributed implementation of the Bellman-Ford algorithm can have **long-lived loops** (of the order of distances) [14]. In fact, the ARPANET initially used this Distributed Bellman-Ford algorithm, but because of long-lived loops, it was replaced in 1979 by a brute-force “link state” algorithm which requires $O(N^2)$ space at each node (to maintain a view of the network topology with a cost for each link).

Since 1979, many new kinds of distance-vector algorithms have been proposed [15, 19, 10, 21, 3, 6, 17, 9] which avoid long-lived loops by using various node coordination mechanisms. For example [15, 19, 10] use diffusion computations [4] to avoid loops entirely. References [21, 3, 17, 9] avoid long-lived loops, but not **short-lived loops**, i.e. loops that disappear in time proportional to $N$ or less. In [21], each node maintains for each destination a set of **paths** (in addition to the distances), one for each of its neighbors. The intention is that the path maintained at node $u$ for a neighbor is the next-hop path of the neighbor with node $u$ appended to the front. Long-lived loops are avoided by not choosing a neighbor as a next-hop if the path maintained for that neighbor contains a loop. However maintaining and exchanging paths is expensive and requires $O(N \times \epsilon \times H)$ storage at each node, where $H$ is the length (in number of links) of a maximum length shortest cost path between any two nodes (note that $H$ can be as high as $N$). References [3, 17, 9]
overcome this problem by having nodes maintain prefinal nodes instead of the paths. The prefinal node for a destination is intended to be the last node before the destination on the next-hop path. Using the prefinal nodes, a node can reconstruct the path to any destination (see Section 6), thereby avoiding long-lived loops.

Understanding distance-vector algorithms, particularly the new ones, is difficult. The analyses in the literature of the above algorithms (e.g. showing that optimal paths are eventually achieved) are operational and incomplete. In this paper, we present a stepwise assertional design of distance-vector algorithms. We go through the following steps:

1. We start our design with the Distributed Bellman-Ford algorithm, referred to as A1. We prove that after any succession of topology changes, the nodes that can still reach the destination eventually achieve and maintain optimal next-hop paths.

2. We next obtain an algorithm, called A2, by adding a path-exchange mechanism to A1. We prove that A2 converges to optimal paths in $O(N)$ steps, assuming synchronous execution of the network; i.e. the routing algorithm executes in steps, and in each step all (and only those) messages that are sent in the previous step are received. This proves that A2 avoids long-lived loops.

3. We next obtain an algorithm, called A3, by adding to A2 a constraint that a node chooses a neighbor as the next-hop for a destination only if the neighbor is also the next-hop for all intermediate destinations on the path to the destination.

4. Our fourth algorithm, called A4, is obtained from A3 by replacing paths with prefinal nodes.

For each algorithm $A_i$, the safety and progress properties satisfied by the previous algorithms hold. In the case of A2 and A3, it is straightforward to check that the proofs for the previous algorithms continue to hold with minor modifications. For A4, we establish that A4 is a well-formed refinement [11] of A3; thus, all safety and progress assertions satisfied by A3 hold for A4 [11].

Many algorithms proposed in the literature use similar mechanisms to algorithms A1 through A4. For example, Old Arpanet Routing Algorithm [14], Routing Information Protocol (an Internet standard) [7], and Inter-Gateway Routing Protocol [8] are variations of A1. Inter-Domain Routing Protocol (ISO draft standard) [18], Border Gateway Protocol (an Internet standard) [12], and the algorithm in [21] are variations of A2. The algorithms in [3, 17, 9] are variations of A4. Hence, understanding the properties of algorithms A1 through A4 is very useful in understanding various internetworking routing protocols. We introduce A3 because showing that properties of A2 hold for A4 is not simple (whereas showing that properties of A2 hold for A3, and properties of A3 hold for A4 is simple).

In section 2, we present our system model and proof rules. In sections 3, 4, 5, 6, we describe A1, A2,
A3, and A4, respectively. In section 7, we give concluding remarks. A preliminary version of algorithms A1 through A4, without most of the analysis, was presented in [1].

2 Preliminaries: System Model and Proof Rules

We use state transition systems and fairness requirements to specify routing protocols, and safety and progress assertions to describe their behaviors (e.g. [11, 20, 13]).

A state transition system consists of a set of state variables, a set of events, and an initial condition on the state variables. The state variables define the set of system states. Each event $e$ is specified by an enabling condition, referred to as enabled($e$) and an (atomically executed) action, referred to as action($e$); together they define a set of state transitions for the event.

A behavior of the state transition system is a sequence of the form $(s_0, f_0, s_1, f_1, \ldots)$, where the $s_i$'s are system states, the $f_i$'s are event names, $s_0$ is an initial state, and for each $i \geq 0$, $(s_i, s_{i+1})$ is a transition of $f_i$. A behavior can be infinite or finite (in which case it ends in a state). In the following definitions, we consider behavior $\sigma = (s_0, f_0, s_1, f_1, \ldots)$.

An event can be subject to a weak fairness. A behavior $\sigma$ satisfies weak fairness for event $e$ iff (1) $\sigma$ is finite and $e$ is not enabled in the last state of $\sigma$, or (2) $\sigma$ is infinite and either $e$ occurs infinitely often or is disabled infinitely often in $\sigma$.

We use two types of safety assertions in this paper: invariant assertions and unless assertions. An invariant assertion is of the form Invariant($A$) where $A$ is a state formula, i.e. a formula which is true or false at each state. By definition, Invariant($A$) holds for a behavior $\sigma$ iff every state $s_i$ in $\sigma$ satisfies $A$.

An unless assertion is of the form $A$ unless $B \lor E$, where $A$ and $B$ are state formulas and $E$ is a set of event names. By definition, $A$ unless $B \lor E$ holds for a behavior $\sigma$ iff for every state $s_i$ in $\sigma$ satisfying $A \land \neg B$, at least one of the following hold: (1) $s_i$ is the last state ($\sigma$ is finite), or (2) $s_{i+1}$ satisfies $A \lor B$, or (3) $f_i$ is in $E$. The event set $E$ can be empty, in which case we simply write $A$ unless $B$.

A safety assertion holds for a state transition system iff it holds for every behavior of the system.

Our progress assertions are of the form $A$ leads-to $B \lor E$, where $A$ and $B$ are state formulas and $E$ is a set of event names. By definition, $A$ leads-to $B \lor E$ holds for a behavior $\sigma$ iff for every $s_i$ in $\sigma$ that satisfies $A$, there is a $j \geq i$ such that $s_j$ is in $\sigma$ and satisfies $B$ or $f_j$ is in $\sigma$ and belongs to $E$. The event set $E$ can be empty, in which case we simply write $A$ leads-to $B$. Given a state transition system and a set of fairness requirements, a leads-to assertion holds for the system iff it holds for every behavior of the system which satisfies the fairness requirements.
We next list the proof rules used in this paper. We use Initial as a state formula specifying the initial condition. Given an event $e$, we use $\{A\}e\{B\}$ to mean the Hoare-triple $\{A \land enabled(e)\}action(e)\{B\}$, i.e., in any state that satisfies $A$, if $e$ is enabled then its occurrence results in a state that satisfies $B$.

**Invariance rule:** $\text{Invariant}(A)$ holds if for some state formula $C$, the following hold:
- $Initial \Rightarrow A$
- for every event $e$, $\{A \land C\}e\{A\}$
- $\text{Invariant}(C)$.

**Implication rule:** $\text{Invariant}(A)$ holds if for some state formula $C$, the following hold:
- $\text{Invariant}(C)$
- $C \Rightarrow A$.

**Unless rule:** A unless $B \lor \mathcal{E}$ holds if for some state formula $C$, the following hold:
- for every event $e \notin \mathcal{E}$, $\{A \land \neg B \land C\}e\{A \lor B\}$
- $\text{Invariant}(C)$.

**Leads-to rule:** $A$ leads-to $B \lor \mathcal{E}$ holds if for some state formula $C$, the following hold:
- for every event $e \notin \mathcal{E}$, $\{A \land \neg B \land C\}e\{A \lor B\}$
- for some event $e$ with weak fairness, $\{A \land \neg B \land C\}e\{B\}$
- $\text{Invariant}(A \land C \Rightarrow enabled(e))$
- $\text{Invariant}(C)$.

**Closure rules:**
- $A$ leads-to $B \lor \mathcal{E}$ holds if $\text{Invariant}(A \Rightarrow B)$ holds.
- $A$ leads-to $B \lor \mathcal{E}$ holds if for some state formula $C$: $A$ leads-to $C \lor \mathcal{E}$ and $C$ leads-to $B \lor \mathcal{E}$ hold.
- $A$ leads-to $B \lor \mathcal{E}$ holds if $A = A_1 \lor A_2$, $A_1$ leads-to $B \lor \mathcal{E}$, and $A_2$ leads-to $B \lor \mathcal{E}$ hold.
- $A \land B$ leads-to $(C \lor (A \land D)) \lor \mathcal{E}$ holds if $A$ unless $C \lor \mathcal{E}$ and $B$ leads-to $D \lor \mathcal{E}$ hold.

These rules are similar to the rules in [13, 2]. It is straightforward to show their soundness (e.g. [11, 20]).

### 3 Algorithm A1

We consider a computer network whose nodes and links form an arbitrary directed graph such that if there is a link from node $u$ to node $v$, then there is a link from node $v$ to node $u$. Let $\text{NODES}$ be the set of nodes, and $\text{LINKS} (\subseteq \text{NODES} \times \text{NODES})$ be the set of links. Node $v$ is a neighbor of node $u$ if $(u, v)$ is a link. Let
neighbors(u) denote the set of neighbors of u. A sequence \( x_0, \ldots, x_n \) of nodes is a path iff \((x_i, x_{i+1})\) is a link for \(0 \leq i < n\). A path is simple if no node is repeated.

A routing protocol is specified by a state transition system and a set of fairness requirements. Each node \(u\) has a set of state variables and a set of events. Each link \((u, v)\) has a state variable, called \(Channel_{uv}\), indicating the sequence of messages in transit. \(Channel_{uv}\) initially equals \(\langle \rangle\), the null sequence. The events of a node can access the state variables of the node, send messages to outgoing links, and receive messages from incoming links. A link \((u, v)\) behaves as a FIFO queue, except when it fails, in which case \(Channel_{uv}\) is set to \(\langle \rangle\); for notational convenience, we group this failure event among the events of node \(u\). We assume that each receive event has weak fairness; this is a convenient way to model finite message propagation delays.

**Conventions:** We use \(u, v, w, x, y, z\) to range over \(NODES\); in some (explicitly stated) cases, they range over \(NODES \cup \{nil\}\). We use \(v, w\) to range over \(neighbors(u)\). We use \(z\) to indicate the destination node. We use \(c, k, d, newcost\) to range over \(I^+ \cup \{0, \infty\}\), indicating a distance or a cost, where \(I^+\) is the set of positive integers. We treat \(\infty\) as a number higher than any number in \(I^+\); e.g. \(\infty + \text{any number is } \infty\).

Given a set \(S\) of numbers, \(\min S\) denotes the smallest number in \(S\). If \(S\) is empty then \(\min S\) returns \(\infty\).

Table 1 specifies the state variables and events of an arbitrary node in \(A1\), the Distributed Bellman-Ford algorithm. (Refer to the table in the following discussion.) Node \(u\) maintains the cost of each outgoing link \((u, v)\) in state variable \(Linkcost_{u,v}(v)\). \(Linkcost_{u,v}(v)\) equals \(\infty\) iff the link is failed; it can change its value at any time due to link failure, link recovery and link cost change events. \(Linkcost_{u,v}(v)\) is never \(0\).

For each destination \(z\), node \(u\) maintains in state variable \(Distvia_{u,v}(v,z)\) an estimate of the distance to \(z\) via neighbor \(v\). It equals \(\infty\) if node \(u\) believes \(z\) cannot be reached via \(v\). The state variable \(Nhop_{u,v}(z)\) indicates the next-hop for destination \(z\). It equals neighbor \(v\) only if \(Distvia_{u,v}(v,z)\) is minimum among all neighbors. \(Nhop_{u,v}(z)\) equals \(nil\) iff \(Distvia_{u,v}(v,z)\) equals \(\infty\) for all neighbors \(v\). Node \(u\) also maintains state variable \(Dist_{u}(z)\) in which it stores the distance via its next-hop, except when \(u = z\) (in which case \(Dist_{u}(z) = 0\)).

Nodes exchange information about their distances to various destinations. Specifically, node \(v\) sends messages of the form \((v, dvector)\), where \(dvector\) is a set of \((z, d)\) pairs such that \(d = Dist_{v}(z)\); note that \(d\) can be \(\infty\).

When \(Linkcost_{u,v}(v)\) changes (either because of link failure, recovery or change in cost), \(Distvia_{u,v}(v,z)\), and if needed \(Nhop_{u}(z)\) and \(Dist_{u}(z)\), is updated for each destination \(z\) (for details see procedure \text{Update}&\text{Send} in table 1). If the distance of any destination \(z\) has been affected (i.e. \(Dist_{u}(z)\) has changed), node \(u\) sends a message to its neighbors containing the \((z, Dist_{u}(z))\) pairs for all affected destinations \(z\).

Additionally, when link \((u, v)\) recovers, \(u\) sends a message to \(v\) containing the \((z, Dist_{u}(z))\) pairs for all
destinations \( z \). This is to ensure that if \( u \) offers a better path for some destination \( z \), node \( v \) will choose \( u \) as its next-hop. This also ensures that if a network becomes disconnected after being disconnected (due to a set of link failures), nodes in different partitions obtain paths to each other.

When node \( u \) receives a \((v, d\_vector)\) message, it updates \( Dist\_v \((v, z)\)\), and if needed \( Nhop_v(z) \) and \( Dist_u(z) \), for each destination \( z \) in \( d\_vector \). If the distance of any destination has been affected, node \( u \) sends a message to its neighbors containing the \((z, Dist_u(z))\) pairs for all affected destinations \( z \).

We say that the network is in a symmetric state if for every link \((u, v)\), link \((u, v)\) is up iff link \((v, u)\) is up. In the rest of this section, we prove that after any succession of topology changes that leaves the network symmetric, for every node \( u \) and every destination \( z \) reachable from \( u \), eventually the next-hop path starting from \( u \) leads to \( z \) and has minimum cost among all paths from \( u \) to \( z \). To specify this formally, we define the following functions (on the system state):

**UPLINKS.** Set of up links. Formally,

\[
= \{(u, v) \in LINKS : \text{Linkcost}_u(v) < \infty \}.
\]

**Symmetric.** Boolean.

\[
= \text{true} \iff (\forall (u, v) \in LINKS : (u, v) \in UPLINKS) \iff (v, u) \in UPLINKS].
\]

**Nhoppath\((u, z)\).** The succession of next-hops for \( z \) starting from \( u \). Formally,

\[
= \{(x_0, \ldots, x_n) \text{ such that } x_0 = u,
\]

\[
\text{for } i \in [0..n-1] : Nhop_{x_i}(z) = x_{i+1}, \land x_i \notin \{x_0, \ldots, x_{i-1}\} \cup \{nil\} \cup \{z\}, \text{ and}
\]

\[
x_n = z \lor Nhop_{x_n}(z) = \text{nill} \lor x_n \in \{x_0, \ldots, x_{n-1}\}.
\]

**Available paths\((u, z)\).** The simple paths from \( u \) to \( z \) over up links. Formally,

\[
= \{(x_0, \ldots, x_n) : x_0 = u \land x_n = z \land \text{[for } i \in [1..n] : (x_{i-1}, x_i) \notin UPLINKS \land x_i \notin \{x_0, \ldots, x_{i-1}\} \text{]} \}.
\]

**Reachable.** Set of node pairs \((u, z)\) such that \( u \) can reach \( z \). Formally,

\[
= \{(u, z) : \text{Available paths}(u, z) \neq \{\} \}.
\]

**Path\_cost\((x_0, \ldots, x_n)\).** The cost of path \((x_0, \ldots, x_n)\). Formally,

\[
= \begin{cases} 
\sum_{i=0}^{n-1} \text{Linkcost}_{x_i}(x_{i+1}) & n > 0 \\
0 & n = 0 \text{ (i.e. path equals } (x_0)) \\
\infty & n < 0 \text{ (i.e. path equals } \{\} \text{)}
\end{cases}
\]

Note that the path cost is \( \infty \) if any link cost in the path is \( \infty \).

**Cost\((u, z)\).** The cost of a minimum cost path from \( u \) to \( z \). Formally,

\[
= \min\{\text{Path\_cost}(p) : p \in \text{Available paths}(u, z)\}.
\]
\[ \text{HighestCost} = \max \{ \text{Cost}(u, z) : (u, z) \in \text{Reachable} \} . \]

\( \text{TC} \). The set of topology change events. Formally,
\[
\text{TC} = \{ \text{LinkFailure}_v(v), \text{LinkRecovery}_c(v, c), \text{LinkCostChange}_u(v, c) : (u, v) \in \text{LINKS} \land c \in T^+ \}.
\]

**Conventions:** We use the term *distance* when we refer to the values of state variables \( \text{Dist}_u(z) \) and \( \text{DistVia}_u(v, z) \), either in the nodes or in transit in the channels. We say "distance \( d \) in transit for destination \( z \)" to mean there is a message in transit whose \( d \)-vector contains a \( (z, d) \) pair. We use the term *cost*, and not "distance", when we refer to the current values of link costs, e.g. \( \text{PathCost} \), \( \text{Cost} \). Note that costs can not change unless a topology change happens.

**Notation:** For any non-empty sequence \( \langle x_0, \ldots, x_n \rangle \), \( \text{last}(\langle x_0, \ldots, x_n \rangle) \) denotes \( x_n \), \( \text{tail}(\langle x_0, \ldots, x_n \rangle) \) denotes \( \langle x_1, \ldots, x_n \rangle \), and \( \text{head}(\langle x_0, \ldots, x_n \rangle) \) denotes \( x_0 \). When applied to a null sequence, \( \text{head}(\langle \rangle) = \text{last}(\langle \rangle) = \text{nil} \) and \( \text{tail}(\langle \rangle) = \langle \rangle \). We use \( @ \) as the concatenation operator for sequences, i.e. \( \langle x_0, \ldots, x_n \rangle @ \langle y_0, \ldots, y_m \rangle = \langle x_0, \ldots, x_n, y_0, \ldots, y_m \rangle \).

We define a boolean function \( \text{HasOptimalPath}(u, z) \) that is true iff the next-hop path starting from \( u \) reaches \( z \) and has optimal cost; note that this implies that all nodes on the next-hop path also have optimal next-hop paths to \( z \). Formally:
\[
\text{HasOptimalPath}(u, z) \equiv \text{last}(\text{NextHopPath}(u, z)) = z \\
\land [\forall x \in \text{NextHopPath}(u, z) : \text{Dist}_x(z) = \text{Cost}(x, z) = \text{PathCost}(\text{NextHopPath}(x, z))]
\]

The desired objective can be stated as follows, where \( A \) is some state formula (that can depend on the routing algorithm):

- \( \text{Symmetric} \land (u, z) \in \text{Reachable} \Rightarrow \text{TC} \lor (u, z) \in \text{Reachable} \land \text{HasOptimalPath}(u, z) \land A \)
- \( \text{Symmetric} \land (u, z) \in \text{Reachable} \land \text{HasOptimalPath}(u, z) \land A \) unless \( \text{TC} \)

That is, after any succession of topology changes that leaves the network in a symmetric state, if there are no further topology changes, then every reachable node \( u \) eventually achieves a stable optimal path to \( z \). We point out that most routing algorithms, including the ones in this paper, do not satisfy the above property if \( A = \text{true} \). That is, it is possible for a node to achieve an optimal next-hop path and then switch to some other non-optimal path. However, eventually, it will find an optimal next-hop path and also satisfy \( A \); once this is achieved, the optimal next-hop path is stable.

The following assertions \( M_1 \) and \( M_2 \) specify an appropriate \( A \) for algorithm A1:

\( (M_1) \) \( \text{Symmetric} \Rightarrow \text{TC} \lor [\forall (u, z) \in \text{Reachable} : \text{HasOptimalPath}(u, z) \land [\forall v \in \text{neighbors}(u) : \text{Channel}_{uv}(z) = \langle \rangle]] \)
optimal path and

\[ A \]

Symmetric assertions of the form /\ holds, then eventually

\[ B \]

Channel(u, v) = \langle m_0, m_1, \ldots, m_n \rangle such that

\[ \exists p_0, \ldots, p_{n+1} : Channel(u, v) = p_0 \otimes \langle m_0 \rangle \otimes p_1 \otimes \langle m_1 \rangle \otimes \ldots \otimes p_n \otimes \langle m_n \rangle \otimes p_{n+1} \land \]

\[ \forall i, 0 \leq i \leq n, \exists d : (z, d) \in m_i \land \]

\[ \forall i, 0 \leq i \leq n + 1, \forall m \in p_i, \forall d : (z, d) \not\in m \].

Theorem 1. A1 satisfies M1 and M2.

Proof of Theorem 1

Readers who are interested in the algorithms but not in the proofs can skip this proof.

Conventions: For a leads-to assertion “A leads-to TC ∨ B”, we refer to A as the left side of the assertion, and B as the right side. We use the same convention for “A unless TC ∨ B” and for “Invariant A ⇒ B”. Most of our leads-to assertions have the form Symmetric ∧ A leads-to TC ∨ B, that is, if Symmetric and A holds, then eventually B holds or a topology change occurs. When informally describing such an assertion, for brevity, we just say “if A holds then eventually B holds”. The same convention is used with assertions of the form “Symmetric ∧ A unless TC ∨ B”. We assume the following precedence of operators: ¬, ∧, ∨, ⇒, Invariant, unless, leads-to. We say cost of a node pair (u, z) and distance of a node pair (u, z) to mean Cost(u, z) and Dist(u, z) respectively.

The following assertions express rather obvious relationship between neighboring nodes:

\( (B_1) \) Dist(u, z) = d ∧ (v, u) ∈ UPLINKS leads-to TC ∨ Distvia(u, v, z) = d + Linkcost(u, v)

\( (B_2) \) Invariant (v, u) ∈ UPLINKS ∧ Channel(u, v) ≠ \{\} ⇒ (z, Dist(u, z)) = last(Channel(u, v))

\( (B_3) \) Invariant (v, u) ∈ UPLINKS ∧ Channel(u, v) = \{\} ⇒ Distvia(u, v, z) = Dist(u, z) + Linkcost(u, v)

\( (B_4) \) Invariant (v, u) ∈ UPLINKS ∧ Distvia(u, v, z) ≠ Dist(u, z) + Linkcost(u, v)

⇒ (z, Dist(u, z)) = last(Channel(u, v))

\( (B_5) \) m in Channel(u, v) leads-to TC ∨ m = front(Channel(u, v))

\( (B_6) \) Channel(u, v) = \langle m \rangle \otimes x leads-to TC ∨ \exists y : Channel(u, v) = \langle x \rangle \otimes y

\( (B_7) \) (z, d) in Channel(u, v) leads-to TC ∨ Distvia(u, v, z) = d + Linkcost(u, v)

\( (B_8) \) (z, d) = front(Channel(u, v)) leads-to TC ∨ Distvia(u, v, z) = d + Linkcost(u, v)
$B_1$, $B_2$, $B_3$, $B_4$, $B_7$ and $B_8$ deal with the distances of neighboring nodes to a destination $z$ and the distances to $z$ in transit between the neighboring nodes.

$B_1$ states that if the distance of $v$ is $d$ and the link $(v, u)$ is not failed, then $u$ eventually learns of $d$. $B_1$ follows from $B_4$ and $B_7$ by the closure.

$B_2$ states that if a channel has distances to $z$, then the last message contains the current distance of the sender. $B_2$ follows from invariance rule.

$B_3$ states that if no distances to $z$ are in transit, then the distance of the receiver through the sender is up-to-date. $B_3$ follows from $B_2$ using invariance rule.

$B_4$ states that if a distance of node $u$ via a neighbor $v$ is not up-to-date, then the current distance of $v$ is in the last message in $Channel_{uv}(z)$. $B_4$ follows from $B_2$ and $B_3$ by implication (left side of $B_4$ implies the negation of the right side of $B_3$; since $B_3$ holds, the left side of $B_3$ must also be false, which implies the left side of $B_2$, which implies the right side of $B_2$, which implies the right side of $B_4$).

$B_5$ states that a message in transit eventually advances to the front of the channel. $B_6$ states that the message in the front of the channel eventually gets removed. $B_5$ follows from leads-to rule (via receive event). $B_5$ follows from $B_6$ by closure. $B_7$ states that each distance in link $(v, u)$ is eventually used to update the distance of $u$ via $v$. $B_8$ states that the distance in the front of a link $(v, u)$ is eventually used to update the distance of $u$ via $v$. $B_8$ follows from leads-to rule (via receive event). $B_7$ follows from $B_3$ and $B_8$ by the closure.

The following safety assertions state that the values of $Symmetric$, $Reachable$, cost of a node pair, and $HighestCost$ do not change. Each of them holds from the unless rule.

(C1) $Symmetric$ unless $TC$

(C2) $Reachable = S$ unless $TC$

(C3) $Cost(u, z) = K$ unless $TC$

(C4) $HighestCost = K$ unless $TC$

We now define functions that, in some sense, characterize the essence of algorithm A1:

In. Maximal subset of $Reachable$ such that $(u, z)$ is a member of $In$ iff
1. $Has_{optimal\_path}(u, z)$,
2. for any message $(x, d)$ in transit, Dist$_w(z)$ is less than $d$,
3. for any node pair $(w, x)$ in $Reachable$ not satisfying $Has_{optimal\_path}(w, x)$,
   
   $Dist_w(z) < Dist_w(x)$ and $Dist_u(z) < Cost(w, x)$.
Out = Reachable - In.

Lowest. The minimum of the cost of node pairs in Out, the distances of node pairs in Out, and the
distances in transit between nodes from which the destination is reachable. Formally,
= \min\{Cost(x, z) : (x, z) \in Out\} \cup
\{Dist_u(z) : (u, z) \in Out\} \cup
\{d : (x, d) \in Channel_{uv} \land (u, x), (v, x) \in Reachable\}.

The intuition behind a node pair \((u, z)\) being in In is the following: \(u\) has an optimal path to \(z\), and this
cannot be affected by any message in transit or by any message that can be generated by other nodes. Note
that if a node pair \((u, z)\) is in In and \(u \neq z\), then \(NHop_u(z) \neq nil\) and the node pair \((NHop_u(z), z)\) is also
in In. If a node pair \((u, z)\) is in In, then the outgoing channels of \(u\) do not contain any \((z, d)\) messages.
This follows from \(B_2\) and the definition of In (i.e., since \((u, z)\) is in In, the messages in transit for \(z\) have larger
distances than the distance of \(u\), and if an outgoing channel of \(u\) contained a message for \(z\), the last message
in that channel for \(z\) would contain a distance which was not larger).

The intuition behind Lowest is the following: Lowest never decreases, and keeps increasing as long
as it is less than HighestCost. Furthermore, Lowest > HighestCost iff In = Reachable (this is because
Lowest > HighestCost means that cost of all reachable node pairs are less than Lowest, hence they are not
in Out). In contrast, the minimum distance in transit can decrease or increase without a change in Out; the
same is true for the minimum distance of a node pair in Out.

We now proceed to prove \(M_1\) and \(M_2\). The proof of \(M_1\) is summarized in Figure 1.

\(M_2\) holds from the unless rule; specifically, once the left side of \(M_2\) holds, no receive event of any node
in Reachable is enabled, and all other events belong to TC. Thus, it suffices to prove \(M_1\).

(M3) Symmetric leads to TC \(\lor\) In = Reachable

\(M_3\) states that eventually In contains all reachable node pairs. \(M_1\) follows from \(M_3\) by closure (since
In = Reachable implies right side of \(M_1\)). Thus it suffices to prove \(M_3\).

(M4) Symmetric leads to TC \(\lor\) Lowest > HighestCost

\(M_4\) states that Lowest eventually exceeds HighestCost. \(M_3\) follows from \(M_4\) by closure. Thus it suffices
to prove \(M_4\).

(M5) Symmetric \(\land\) Lowest \(\leq\) HighestCost leads to TC \(\lor\) Lowest \(\geq\) \(k + 1\)

\(M_4\) follows from \(M_5\), \(C_1\) and \(C_4\) by closure. Thus it suffices to prove \(M_5\). We first define the following
functions:
Figure 1: Proof of $M1$. Each arrow indicates that the tail assertion is used in the proof of the head assertion. Proof rule used is indicated in parenthesis.

$DV_{ia}(k) = \{(u,v,z) : Dist_{via}(v,z) = k \land (v,z) \in Out\}$

$DTransit(k) = bag\{(u,v,z) : (z,k) \in Channel_{uv} \land (u,v),(v,z) \in Reachable\}$

Note that $DTransit(k)$ is a bag; i.e. if there are two messages whose distance vectors contain the same $(z,k)$ pair in the same channel, $DTransit(k)$ contains two $(u,v,z)$ triplets.

We next define the following assertions:
(M₆) Symmetric ∧ Lowest = k leads to TC ∨ Lowest ≥ k ∧ |DVia(k)| = 0

(M₇) Symmetric ∧ Lowest ≥ k ∧ |DVia(k)| = 0 leads to TC ∨ Lowest ≥ k ∧ |DVia(k)| = 0 ∧ |DTransit(k)| = 0

(M₈) Invariant Lowest ≥ k ∧ |DVia(k)| = 0 ∧ |DTransit(k)| = 0 ⇒ Lowest > k

M₆ and M₇ state that if Lowest = k, then DVia(k) and DTransit(k) eventually become empty. At that point, M₈ states that Lowest is greater than k. M₅ follows from M₆, M₇, M₈ and C₁ by closure.

Thus it suffices to prove M₆, M₇ and M₈, which is done next.

Proof of M₈

The following assertions state that if Lowest ≥ k and DVia(k) and DTransit(k) are empty, node pairs in Out have both costs and distances higher than k.

(M₉) Invariant Lowest ≥ k ∧ |DVia(k)| = 0 ∧ |DTransit(k)| = 0 ∧ Cost(u, z) = k ⇒ (u, z) ∈ In

(M₁₀) Invariant Lowest ≥ k ∧ |DVia(k)| = 0 ∧ (u, z) ∈ Reachable ∧ ¬Has_optimal_path(u, z) ⇒ Distₜ(z) > k

M₈ follows from M₉ and M₁₀ by implication. (The details are as follows: From M₉, the cost of a node pair in Out is greater than k. From |DTransit(k)| = 0 and Lowest ≥ k, the minimum distance in transit is greater than k. From M₁₀, the distance of a node pair in Out is greater than k; note that if (u, z) is in Out and Has_optimal_path(u, z), then Distₜ(z) = Cost(u, z) > k. Hence Lowest is greater than k.)

Thus it suffices to prove M₉ and M₁₀. We next proceed to prove M₉.

(M₁₁) Invariant Lowest ≥ k ∧ |DVia(k)| = 0 ∧ Cost(u, z) = k ⇒ Has_optimal_path(u, z)

M₉ follows from M₁₁ and M₁₀ by implication. (The details are as follows: Consider a node pair (u, z) satisfying left side of M₉. (u, z) has optimal path (from M₁₁). Distances of node pairs (w, x) not satisfying Has_optimal_path(w, x) are greater than k (from M₁₀). Costs of node pairs (w, x) not satisfying Has_optimal_path(w, x) are greater than k (from M₁₁ and Lowest ≥ k). Distances in transit are greater than k (from Lowest ≥ k and |DTransit(k)| = 0.)

Thus it suffices to prove M₁₁ and M₁₀. We next proceed to prove M₁₁.

(M₁₂) Invariant u ≠ z ⇒ Distₜ(z) = \{DistViaᵤ(v, z) : v ∈ neighbors(u)\}

(M₁₃) Invariant \[\forall(v, z) ∈ In ∧ v ∈ neighbors(u) : DistViaᵤ(v, z) ≥ Cost(u, z)\]

(M₁₄) Invariant Cost(u, z) ≤ Lowest ∧ u ≠ z ⇒ \[\exists(v, z) ∈ In ∧ v ∈ neighbors(u) : DistViaᵤ(v, z) = Cost(u, z)\]

M₁₂ states that distance of a node pair equals the minimum of distances via its neighbors. M₁₄ follows from invariance rule.
M_{13} states that distance of a node via a node pair in \( In \) is greater than or equal to the cost of the node. \( M_{13} \) follows from \( B_7 \) and \( B_3 \) by implication (since \((v, z)\) is in \( In \), \( Dist_{v, z}(z) = \text{Cost}(v, z) \) and \( v \)'s outgoing channels do not contain a message for \( z \), hence \( \text{Dist}_{via}(v, z) \) equals \( \text{Cost}(v, z) + \text{Linkcost}(v) \), which is greater than or equal to cost of node pair \((u, z)\)).

\( M_{14} \) states that a node pair with cost less than or equal to \( \text{Lowest} \) has a neighbor in \( In \) and its distance via this neighbor equals its cost. \( M_{14} \) follows from \( B_2 \) and \( B_3 \) by implication (note that if cost of \((u, z)\) is less than or equal to \( \text{Lowest} \) and \( v \) is \( u \)'s next node on an optimal path, then \((v, z)\) is in \( In \) since \( v \) has a smaller cost; also since outgoing channels of \( v \) do not contain a message for \( z \), the distance of \( u \) via \( v \) equals cost of \((u, z)\)).

\( M_{11} \) follows from \( M_{14}, M_{13} \) and \( M_{12} \) by implication. (The details are as follows: Consider node pair \((u, z)\) that satisfies left side of \( M_{11} \). From \( M_{14} \), there is a neighbor \( v \) of \( u \) such that \((v, z)\) is in \( In \) and \( u \)'s distance to \( z \) via \( v \) equals its cost \( k \). From \( M_{13} \), \( u \)'s distance to \( z \) via neighbors in \( In \) is not less than \( k \). From \( |Dvia(k)| = 0 \) and \( \text{Lowest} \geq k \) (left side of \( M_{11} \)), \( u \)'s distance to \( z \) via neighbors in \( Out \) is higher than \( k \). Thus from \( M_{12} \), \( u \)'s distance to \( z \) is \( k \) and \( N\text{hop}_u(z) \) is a neighbor \( v \) in \( In \). Thus \( u \) has optimal path.)

Thus it suffices to prove \( M_{18} \). \( M_{18} \) follows from \( M_{11}, M_{12} \) and \( M_{13} \) by implication. (The details are as follows: Consider a node pair \((u, z)\) that satisfies left side of \( M_{18} \). From \( \neg \text{Has-optimal-path}(u, z) \) and \( \text{Lowest} \geq k \), we have \( \text{Cost}(u, z) \geq k \). From \( \neg \text{Has-optimal-path}(u, z) \) and \( M_{11} \), we have \( \text{Cost}(u, z) \neq k \). Thus, \( \text{Cost}(u, z) \geq k \). From \( M_{18} \), \( u \)'s distance to \( z \) via neighbors in \( In \) is greater than \( k \). From \( |Dvia(k)| = 0 \) and \( \text{Lowest} \geq k \) (left side of \( M_{18} \)), \( u \)'s distance to \( z \) via neighbors in \( Out \) is greater than \( k \). Thus, from \( M_{12} \), \( u \)'s distance to \( z \) is greater than \( k \).

This completes the proof of \( M_8 \).

**Proof of \( M_6 \)**

We repeat \( M_6 \):

\[
(M_6) \quad \text{Symmetric} \land \text{Lowest} \geq k \implies \text{TC} \lor \text{Lowest} \geq k \land |Dvia(k)| = 0
\]

Define

\[
(M_{15}) \quad \text{Symmetric} \land \text{Lowest} \geq k \land |Dvia(k)| = n > 0 \implies \text{TC} \lor |Dvia(k)| = n - 1
\]

\[
(M_{16}) \quad \text{Lowest} \geq k \quad \text{unless} \quad \text{TC}
\]

\( M_{16} \) follows from the unless rule. \( M_6 \) follows from \( M_{15}, M_{16} \) and \( C_1 \) by closure. Thus it suffices to prove \( M_{15} \).

\[
(M_{17}) \quad \text{Lowest} \geq k \land |Dvia(k)| \leq n \quad \text{unless} \quad \text{TC}
\]
M_{17} states that if \( \text{Lowest} \geq k \) then the size of \( D\text{Via}(k) \) does not increase. \( M_{17} \) follows from the unless rule.

\( M_{15} \) follows from \( M_{17} \) and \( B_1 \) by closure. (The details are as follows: From \( B_1 \), \( \text{Distvia}_u(v, z) \) eventually becomes greater than \( \text{Lowest} \) (since \( d \) in \( B_1 \) is greater than \( \text{Lowest} \)), hence decreases \( |D\text{Via}(k)| \). \( M_{17} \) ensures that \( |D\text{Via}(k)| \) does not increase before \( \text{Distvia}_u(v, z) \) becomes greater than \( \text{Lowest} \).)

This completes the proof of \( M_6 \).

Proof of \( M_7 \)

We repeat \( M_7 \):

\[
(M_7) \quad \text{Symmetric} \land \text{Lowest} \geq k \land |D\text{Via}(k)| = 0 \quad \text{leads-to} \quad \text{T}C \lor \text{Lowest} \geq k \land |D\text{Via}(k)| = 0 \land |D\text{Transit}(k)| = 0
\]

Define

\[
(M_{18}) \quad \text{Symmetric} \land \text{Lowest} \geq k \land |D\text{Via}(k)| = 0 \land |D\text{Transit}(k)| = n > 0 \quad \text{leads-to} \quad \text{T}C \lor |D\text{Via}(k)| = 0 \land |D\text{Transit}(k)| = n - 1
\]

\( M_{18} \) states that if \( \text{Lowest} \geq k \) and \( D\text{Via}(k) \) is empty, then the size of \( D\text{Transit}(k) \) eventually decreases. \( M_7 \) follows from \( M_{18}, M_{19} \) and \( C_1 \) by closure. Thus it suffices to prove \( M_{18} \).

\[
(M_{19}) \quad \text{Lowest} \geq k \land |D\text{Via}(k)| = 0 \land D\text{Transit}(k) \text{ bag-subset } S \text{ unless } \text{T}C
\]

\( M_{19} \) states that if \( \text{Lowest} \geq k \) and \( D\text{Via}(k) \) is empty then \( D\text{Transit}(k) \) does not expand\(^1\). \( M_{19} \) follows from unless rule.

\( M_{18} \) follows from \( B_5, B_6 \) and \( M_{19} \) by closure. (The details are as follows: From \( B_5 \), a message participating in \( D\text{Transit}(k) \) advances to front. From \( B_6 \), it gets removed, decreasing \( |D\text{Transit}(k)| \). \( M_{19} \) ensures that \( D\text{Transit}(k) \) does not expand while the message advances to front.)

This completes the proof of \( M_7 \), and hence of Theorem 1.

End of proof of Theorem 1

Even though we have shown that after any succession of topology changes, the nodes that can reach the destination obtain optimal paths, this convergence may contain long-lived loops and be very lengthy. For example, consider the simple network in Figure 2.a. Three are three nodes \( u, v, \) and \( z \). Destination node is \( z \). Assume all link costs are 1. Numbers on the arrows indicate the distances of nodes via their neighbors, and solid arrows indicate the next-hops to \( z \). That is, node \( u \)'s distance to \( z \) via \( z \) is 1 and via node \( v \) is 3. In Figure 2.b, cost of the link \( (u, z) \) increases to \( D \) such that \( D > 3 \). As a result \( u \) chooses \( v \) as its

\(^1\) Bag \( S \) is a bag-subset of bag \( Z \) if every element \( m \) of \( S \) is also an element of \( Z \). Note that, if \( S \) contains \( k \) instances of \( m \), then \( Z \) contains at least \( k \) instances of \( m \).
next-hop, causing a loop, and sends its new distance to \( v \). Upon receiving this message, \( v \) will add 1 (i.e. \( \text{Linkcost}_v(u) \)) to this distance (see Figure 2.c), and send a message back to \( u \), causing node \( u \)'s distance to increase (see Figure 2.d). Node \( u \) and \( v \) will keep on exchanging messages (referred to as \textit{bouncing effect} in the literature), each time increasing their distances by 2 (i.e. \( \text{Linkcost}_u(v) + \text{LinkCost}_v(u) \)) until node \( u \)'s distance via \( v \) exceeds \( D \), at which point \( u \) chooses \( z \) as its next-hop. This convergence will require \((D-1)/(\text{Linkcost}_u(v) + \text{LinkCost}_v(u))\) number of exchanges. Note that if \( D \) is \( \infty \) (i.e. link \((u, z)\) fails, and thus nodes \( u \) and \( v \) cannot reach \( z \)), they will exchange distances indefinitely (referred to as \textit{count-to-infinity problem}). With more realistic network topologies, this behavior can be even more complex, for example: loops can involve multiple hops and breaking one loop may cause another loop. In the next algorithm, these problems are avoided.

4 Algorithm A2

Table 2 specifies the state variables and events of a node in \textbf{A2}. (Refer to the table in the following discussion.) Each node maintains the state variables required for \textbf{A1}. In addition, node \( u \) maintains in state variable \( \text{Routevia}_u(v, z) \) an estimate of the next-hop path for destination \( z \) via neighbor \( v \). It is equal to the null sequence if node \( u \) believes \( z \) cannot be reached via \( v \). Node \( u \) also maintains a state variable \( \text{Costseq via}_u(v, z) \) which stores the sequence of estimated link costs for the corresponding links in \( \text{Routevia}_u(v, z) \). State variables \( \text{Route}_u(z) \) and \( \text{Costseq}_u(z) \) store the route and cost sequence via node \( u \)'s next-hop.

\textbf{Convention: } We use the term \textit{route} to refer to estimates maintained by nodes of next-hop paths.

The variables \( \text{Costseq}_u(z) \) and \( \text{Costseq via}_u(v, z) \) are auxiliary variables; they are needed for verification only, and do not have to be implemented. (Formally they satisfy the following conditions: (1) they do not
affect the enabling condition of any event, and (2) they do not affect the update of any nonauxiliary state variable [16].

Algorithm A2 is like algorithm A1, except that A2 uses paths to avoid long-lived loops. Long-lived loops in the next-hop path for destination $z$ can be avoided by having node $u$ not choose a neighbor $v$ as its next-hop if $Route_{via_u}(v, z)$ contains a cycle. Another way to achieve the same effect is by having node $v$ send $\infty$ as its distance to node $u$ if node $u$ is in $Route_v(z)$. We have chosen the second approach, as specified in the last five lines of procedure $Update\&Send$ in Table 2. That is, sending $\infty$ as the distance prevents the receiver from choosing a route with a loop. It does not prevent the receiver from choosing an optimal path.

In addition to exchanging distances, nodes also exchange information about their paths and cost sequences. More precisely, node $v$ sends messages of the form $(v, d_{vector})$, where $d_{vector}$ is a set of $(z, d, p, cs, rd)$ tuples such that either (1) $d = rd = Dist_v(z)$, $p = Route_v(z)$, and $cs = Costseq_v(z)$ if node $u$ is not in $Route_u(z)$, or (2) $d = \infty$, $p = \emptyset$, $cs = \emptyset$, and $rd = Dist_v(z)$, if node $u$ is in $Route_u(z)$. Fields $cs$ and $rd$ are auxiliary fields, and do not have to be implemented ($rd$ is only used in the proof of Theorem 2).

When $Linkcost_u(v)$ changes, $Distvia_u(v, z)$, $Routevia_u(v, z)$, $Costseqvia_u(v, z)$, and if needed $Nhop_u(z)$, $Dist_u(z)$, $Route_u(v, z)$, and $Costseq_u(z)$, are updated for each destination $z$ (as shown in procedure $Update\&Send$ in Table 2). If the distance or route of any destination has been affected, node $u$ sends messages to its neighbors for all affected destinations $z$ (as described in the previous paragraph).

When node $u$ receives a $(v, d_{vector})$ message, it updates its state variables for each destination $z$ in $d_{vector}$ (note that $rd$ is not used to update any state variable). If the distance or route of any destination has been affected, node $u$ sends messages to its neighbors.

**Theorem 2.** A2 satisfies $M_1$ and $M_2$.

**Proof of Theorem 2**

The proof of Theorem 2 is identical to that of Theorem 1, except that the assertions $B_1$-$B_8$ describing the relationship between neighboring nodes, are replaced by new assertions $B_1$-$B_9$ below.

The main differences between A2 and A1 are reflected in the new $B$ assertions. First, messages in transit may contain $\infty$ as distance even though the sender’s distance is finite (see $B_2$ below). This only happens when the receiver is on the sender’s route. Second, when the channel between two nodes do not contain a distance for a destination, distance of the receiver via the sender may not equal the sum of sender’s distance and the cost of the link between them (see $B_3$ below). $B_4$ now has two parts $B_{4a}$ and $B_{4b}$; the first part covers the case when the receiver is not on the route of the sender, and the second part covers the case when
it is. Proofs of \(B_1\)–\(B_9\) are identical to their counterparts in \(A_1\). (Assertions \(B_5\) and \(B_6\) stay the same.)

\[(B_1)\] \(\text{Dist}_u(z) = d \land \text{Route}_u(z) = p \land (v, u) \in \text{UPLINKS}\)
leads to \(TC \lor (u \not\in p \Rightarrow \text{Dist} \chi_{\text{via}}(v, z) = d + \text{Linkcost}_u(v)) \lor (u \not\in p \Rightarrow \text{Dist} \chi_{\text{via}}(v, z) = \infty)\)

\[(B_2)\] Invariant \(\text{Channel}_{vu}(z) \neq \emptyset \land (u, v) \in \text{UPLINKS}\)
\[\Rightarrow ((z, \text{Dist}_u(z), \text{Route}_u(z), \text{Costseq}_u(z), \text{Dist}_v(z)) = \text{last}(\text{Channel}_{vu}(z)) \land u \not\in \text{Route}_e(z)) \lor ((z, \infty, \emptyset, \emptyset, \text{Dist}_v(z)) = \text{last}(\text{Channel}_{vu}(z)) \land u \in \text{Route}_e(z))\)

\[(B_3)\] Invariant \(\text{Channel}_{vu}(z) = \emptyset \land (u, v) \in \text{UPLINKS}\)
\[\Rightarrow (\text{Dist} \chi_{\text{via}}(v, z) = \text{Dist}_u(z) + \text{Linkcost}_u(v) \land u \not\in \text{Route}_e(z)) \lor (\text{Dist} \chi_{\text{via}}(v, z) = \infty \land u \in \text{Route}_e(z))\)

\[(B_{1a})\] Invariant \((u, v) \in \text{UPLINKS} \land \text{Dist} \chi_{\text{via}}(v, z) \neq \infty \land u \not\in \text{Route}_e(z)\)
\[\Rightarrow (z, \text{Dist}_u(z), \text{Route}_u(z), \text{Costseq}_u(z), \text{Dist}_v(z)) = \text{last}(\text{Channel}_{vu}(z))\)

\[(B_{1b})\] Invariant \((u, v) \in \text{UPLINKS} \land \text{Dist} \chi_{\text{via}}(v, z) \neq \infty \land u \in \text{Route}_e(z)\)
\[\Rightarrow (z, \infty, \emptyset, \emptyset, \text{Dist}_v(z)) = \text{last}(\text{Channel}_{vu}(z))\)

\[(B_7)\] \((z, d, p, cs, rd)\) in \(\text{Channel}_{vu}\) leads to \(TC \lor \text{Dist} \chi_{\text{via}}(v, z) = d + \text{Linkcost}_u(v)\)

\[(B_8)\] \((z, d, p, cs, rd) = \text{front}(\text{Channel}_{vu})\) leads to \(TC \lor \text{Dist} \chi_{\text{via}}(v, z) = d + \text{Linkcost}_u(v)\)

\[(B_9)\] Invariant \((z, d, p, cs, rd)\) in \(\text{Channel}_{vu}\) \(\Rightarrow d = \infty \lor d = rd\)

\(B_9\) states that \(rd\) in a message is less than or equal to the corresponding \(d\). \(B_9\) follows from invariance rule.

We redefine \(\text{In}\), \(\text{Lowest}\) and \(\text{DTransit}\) for \(A_2\) as follows:

\(\text{In}\). Maximal subset of \(\text{Reachable}\) such that \((u, z)\) is a member of \(\text{In}\) iff

1. \(\text{HasOptimalPath}(u, z)\),
2. for any message \((x, d, p, cs, rd)\) in transit, \(\text{Dist}_u(z)\) is less than \(rd\),
3. for any node pair \((w, x)\) in \(\text{Reachable}\) not satisfying \(\text{HasOptimalPath}(w, x)\),
   \[\text{Dist}_u(z) < \text{Dist}_w(x)\] and \(\text{Dist}_u(z) < \text{Cost}(w, x)\).

\(\text{Lowest}\). Formally,
\[= \min\{\text{Cost}(x, z) : (x, z) \in \text{Out}\} \cup \{\text{Dist}_u(z) : (u, z) \in \text{Out}\} \cup \{rd : (x, d, p, cs, rd) \in \text{Channel}_{vu} \land (u, x), (v, x) \in \text{Reachable}\}\).
\(\text{DTransit}(k) = \text{bag}\{(u, v, z) : (z, d, p, cs, rd) \in \text{Channel}_{vu} \land rd = k \land (u, z), (v, z) \in \text{Reachable}\}\).
The proof of Theorem 2 is identical to the proof of Theorem 1 with new $B$ assertions. $B_9$ is required for $M_{17}$, $M_{18}$ and $M_{19}$ to hold for $A_2$. Except for these changes, every assertion used in the proof of Theorem 1 also holds for $A_2$ (and the proof is identical). Hence $M_1$ and $M_2$ hold for $A_2$.

**End of proof of Theorem 2**

Next, we establish that after any succession of topology changes that leaves the network symmetric, $A_2$ achieves optimal paths within $N + H$ steps assuming synchronous execution.

We define a **synchronous execution** as follows: Each message includes a *step counter* which is a non-negative integer. Any message sent by a receive event has step counter one higher than the step counter of the received message. Any topology change event sets the step counter of all messages (including the ones being generated) to zero. We require that *Receive* events are executed such that the sequence of step counters of the received messages is non-decreasing. Formally, we define *Step* to be the step counter of the last message received, and add the following $SE$ condition as a conjunct to the enabling condition of every receive event:

$$SE : \text{step counter of the message to be received} = \text{minimum step counter of the messages in transit}$$

Note that *Step* equals 0 immediately after any topology change.

The following assertions $N_1$ and $N_2$ state the desired property, that is, reachable node pairs achieve optimal paths within $N + H$ steps, and other node pairs obtain $\infty$ distances within $N$ steps.

$$\begin{align*}
(N_1) \text{Symmetric} \land \text{Step} = 0 & \text{ leads-to } TC \lor \text{Step} \leq N + H \land \\
\forall (u, z) \in \text{Reachable : Has optimal path}(u, z) \land \forall v \in \text{neighbors}(u) : \text{Channel}_{uv}(z) = \emptyset \\
(N_2) \text{Symmetric} \land \text{Step} = 0 & \text{ leads-to } TC \lor \text{Step} \leq N \land \forall (u, z) \notin \text{Reachable : Dist}_u(z) = \infty
\end{align*}$$

**Theorem 3.** Assuming synchronous execution, $A_2$ satisfies $N_1$ and $N_2$.

**Proof of Theorem 3**

The rest of Section 4 is a proof of Theorem 3. Readers interested in the algorithms but not in the proofs can skip to Section 5.

**Conventions:** We use $\text{step}# \#$ to refer to the step number of a message.

We recast the assertions relating the states of neighbor nodes assuming synchronous execution:

$$\begin{align*}
(D_1) \text{Step} = n \land (v, u) \in \text{UPLINKS} \land \text{Dist}_u(z) = d \land \text{Route}_u(z) = p \land \text{Costseq}_u(z) = cs \land u \notin p & \text{ leads-to } TC \lor \text{Step} \leq n + 1 \land \text{Distvia}_u(v, z) = d + \text{Linkcost}_u(v) \\
& \land \text{Routevia}_u(v, z) = (u) # p \land \text{Costseqvia}_u(v, z) = (\text{Linkcost}_u(v)) \# cs
\end{align*}$$

18
that if $u$ does not distinguish between up and down links.

Suppose link $(v, u)$ is not failed. Given any state of $v$’s distance, route and cost sequence to $z$, $D_1$ states that if $u$ is not on the route from $v$ to $z$, then $u$ eventually learns of $v$’s state within one step. $D_2$ states that if $u$ is on the route from $v$ to $z$, then $u$ eventually learns within one step that $v$ has a distance of $\infty$, route of $\{\}$, and cost sequence of $\{\}$. $D_3$ and $D_4$ (and $D_5$ and $D_6$) make the same distinction. $D_3$ and $D_4$ follow from invariance rule. $D_5$ and $D_6$ follows from the leads-to rule (via receive event). $D_1$ follows from $D_3$, $B_5$, and $D_5$ by closure. $D_2$ follows from $D_4$, $B_5$, and $D_6$ by closure.

Define $\langle x_0, \ldots, x_n \rangle$ to be a ud-path from $x_0$ to $x_n$ if $\forall 0 \leq i < n : (x_i, x_{i+1}) \in \text{LINKS}$. Note that ud-path does not distinguish between up and down links.

Some safety assertions:

$(E_1)$ Invariant $\text{Route}_u(z)$ is a simple ud-path

$(E_2)$ Invariant $|\text{Route}_u(z)| \leq N \land |\text{Route}_u(z)| \leq N \land |(z, d, p, cs, rd)\text{ in transit} \Rightarrow |p| \leq N$
\((E_3)\) Invariant \(\text{Dist}_{via_u}(v, z) = \sum \{c : c \in \text{Cost}_{seq_{via_u}}(v, z)\}\)
\(\land \text{Dist}_u(z) = \sum \{c : c \in \text{Cost}_u(z)\}\)
\(\land [(z, d, p, cs, rd) \text{ in transit } \Rightarrow d = \sum \{c : c \in cs\}]\)

\(E_1\) follows from invariance rule. \(E_2\) states that route lengths (in number of links) are bounded above by \(N\). \(E_3\) follows from \(E_1\) by implication (since a simple path may contain at most \(N\) nodes).

\(E_3\) states that all distances equal the sum of the link costs in the corresponding cost sequences (we assume \(\sum \{\} = \infty\)). \(E_3\) follows from invariance rule.

We define the following:

**Consistent_distances.** Boolean function. True iff (1) distance of any node pair equals path cost of its route, (2) distance of any node pair via a neighbor equals path cost of its route via that neighbor, and (3) any distance in transit in a message equals path cost of the route in the same message. Formally,
\[= \left[\forall u, z \in \text{NODES} : \text{Dist}_u(z) = \text{PathCost(Route}_u(z))\right]\]
\(\land \left[\forall u, z \in \text{NODES}, \forall v \in \text{neighbors}(u) : \text{Dist}_{via_u}(v, z) = \text{PathCost(Route}_{via_u}(v, z))\right]\]
\(\land \left[\forall (z, d, p, cs, rd) \text{ in transit } : d = \text{PathCost}(p)\right].\]

**Done.** Set of node pairs. Formally,
\[= \{(u, z) \in \text{Reachable} : \left[\forall x \in \text{Route}_u(z) : \text{HasOptimalPath}(x, z) \land \left[\forall v \in \text{neighbors}(x) : \text{Channel}_{xv}(z) = ()\right]\right]\}\]

The proof of \(N_1\) is summarized in Figure 3. \(A2\) achieves \(N_1\) in two stages: first within \(N\) steps \(\text{Consistent_distances}(z)\) is established; after that within \(H\) steps \(\text{Done} = \text{Reachable}\) is established (which implies the right side of \(N_1\)). Formally,
\((N_3)\) Symmetric \(\land\) Step = 0 \(\Rightarrow\) TC \(\lor\) Step \(\leq\) \(N\) \(\land\) Consistent_distances

\((N_4)\) Symmetric \(\land\) Step = \(j\) \(\land\) Consistent_distances
\(\quad\Rightarrow\) TC \(\lor\) Step \(\leq\) \(j + H\) \(\land\) Done = Reachable

\(N_1\) follows from \(N_3\), \(N_4\) and \(C_1\) by closure. \(N_2\) follows from \(N_3\) by closure. Thus it suffices to prove \(N_3\) and \(N_4\).

**Proof of \(N_3\)**

We define the following:

\(k_{\text{Consistent}}(\langle x_0, \ldots, x_n \rangle, \langle c_0, \ldots, c_n \rangle)\). Boolean function where \(\langle x_0, \ldots, x_n \rangle\) is a ud-path and \(c_i\)'s are costs.

True iff the link costs of the first \(k\) links \((x_i, x_{i+1})\) equal respectively the first \(k\) costs \(c_i\). Formally,
\[= \text{true} \iff i \in [0, \ldots, \min(k - 1, n - 1)] : c_i = \text{LinkCost}_{x_i}(x_{i+1}).\]
Figure 3: Proof of $N_1$.

$k\ Consistent\ Routes$. Boolean function. True iff all routes are $k$-consistent. Formally,
= [\forall u, z \in NODES : k\_Consistent(Route_u(z), Costseq_u(z))]
\land [\forall u, z \in NODES, v \in neighbors(u) : k\_Consistent(Routevia_u(v, z), Costseqvia_u(v, z))]
\land [\forall (z, d, p, cs, rd) in transit : k\_Consistent(p, cs)].

Note that the first argument of \(k\_Consistent\) in the definition of \(k\_Consistent\_Routes\) is a \(ud\)-path (from \(E_1\)).

\((N_5)\) Symmetric \land Step = 0 \ leads-to \ TC \lor Step \leq N \land N\_Consistent\_Routes

\(N_5\) states that within \(N\) steps all routes are \(N\)-consistent. \(N_3\) follows from \(N_5, E_2\) and \(E_3\) by closure. (For the \(Routevia\) part of \(Consistent\_Distances\), the details are as follows: From \(E_3\), \(Distvia_u(v, z) = \sum\{c : c \in Costseqvia_u(v, z)\}\). From \(N\_Consistent\_Routes\) (right side of \(N_5\)) and \(|Routevia_u(v, z)| \leq N\) \((E_2)\), we have that \(Distvia_u(v, z)\) equals the current cost of \(Routevia_u(v, z)\). Thus it suffices to prove \(N_5\).

\((N_6)\) Symmetric \land Step \leq k \land k\_Consistent\_Routes
\leads-to \ TC \lor Step \leq k + 1 \land (k + 1)\_Consistent\_Routes

\((N_7)\) Symmetric \land k\_Consistent\_Routes \ unless \ TC

\(N_6\) state that if at step \(k\) all routes are \(k\)-consistent, then within one more step all routes are \((k + 1)\)-consistent. \(N_7\) state that once \(k\_Consistent\_Routes\) is established, it continues to hold. \(N_7\) follows from the unless rule. \(N_5\) follows from \(N_6, N_7\) and \(C_1\) by closure (since \(0\)-consistency is true for any route at any step).

Thus it suffices to prove \(N_6, N_6\) follows from \(D_1\), and \(D_2\) by closure. To see this, suppose the route and the cost sequence of a node \(v\) are \(p\) and \(cs\), respectively \((p\) and \(cs\) are \(k\)-consistent\). Then in at most one step, the route and the cost sequence of a neighbor \(u\) via \(v\) either become \((u)@p\) and \((Linkcost_u(v))@cs\), or become {} and {}. In either case, they are \((k + 1)\)-consistent.

**Proof of \(N_4\)**

We first define the following:

\(depth(u, z)\). Minimum length (in number of links) of a minimum cost path from \(u\) to \(z\). Formally,

\[\text{depth}(u, z) = \min\{|p| : p \in Available\_paths(u, z) \land Path\_cost(p) = Cost(u, z)\}\].

Note that \(depth(z, z) = 0\) since \(|\{z\}| = 0\).

\(k\_Reachable\). Subset of node pairs in \(Reachable\) with depth less than or equal to \(k\). Formally,

\[\{ (u, z) \in Reachable : depth(u, z) \leq k \}\].

Some safety assertions:

\((N_8)\) \(depth(u, z) = k\) \ unless \(TC\)
$N_8$ states that the value of $\text{depth}(u, z)$ does not change. $N_8$ follows from the unless rule.

(N) $\text{Symmetric} \land \text{Consistent distances} \land \text{unless TC}$

$N_9$ follows from the unless rule.

(N) $\text{Symmetric} \land \text{Step} = j \land \text{Consistent distances} \land \text{unless TC}$

$N_{10}$ states that once consistent distances are obtained, within $k$ steps, Done will contain all nodes in $k_{\text{Reachable}}$. $N_4$ follows from $N_{10}$ by replacing $k$ by $H$ (note that $H_{\text{Reachable}} = \text{Reachable}$ from the definition of $H$). Thus it suffices to prove $N_{10}$.

(N) $\text{Symmetric} \land \text{Step} = j \land \text{Consistent distances} \land \text{unless TC}$

leads-to $TC \lor \text{Step} \leq j + 1 \land (k + 1)_{\text{Reachable}} \subseteq \text{Done}$

(N) $\text{Invariant Step} > 0 \Rightarrow (z, z) \in \text{Done}$

$N_{11}$ states that once consistent distances are obtained and Done contains all nodes in $k_{\text{Reachable}}$, within one step Done will contain all nodes in $(k + 1)_{\text{Reachable}}$. $N_{11}$ states that Done includes $0_{\text{Reachable}}$ after all messages generated by topology change events are received (at this time, outgoing channels of $z$ do not contain any message for destination $z$). $N_{11}$ follows from $N_9$, $N_{11}$, $N_{12}$, and $N_8$ by closure. $N_{11}$ follows from $E_1$ using the invariance rule (from $E_1$, a message received by $z$ does not contain a distance for $z$, hence $z$ always has an optimal path).

(N) $\text{Symmetric} \land \text{Step} = j \land \text{Consistent distances} \land \text{unless TC}$

leads-to $TC \lor \text{Step} \leq j + 1 \land (u, z) \in \text{Done}$

(N) $\text{Invariant Symmetric} \land \text{Consistent distances} \land \text{unless TC}$

$N_{13}$ states that once consistent distances are obtained and done contains all nodes in $k_{\text{Reachable}}$, within one step a node $u$ in $\text{Reachable}$ with depth $k + 1$ will join Done. $N_{14}$ states that once consistent distances are obtained, Done does not shrink. $N_{14}$ follows from the unless rule. $N_{11}$ follows from $N_9$, $N_{13}$, $N_{14}$, and $C_1$ by closure. Thus it suffices to prove $N_{13}$.

(N) $\text{Invariant Symmetric} \land \text{Consistent distances} \land \text{unless TC}$

leads-to $TC \lor \text{Step} \leq j + 1 \land (u, z) \in \text{Done}$

(N) $\text{Invariant Symmetric} \land \text{Consistent distances} \land \text{unless TC}$

$N_{15}$ states that if consistent distances are obtained and $\text{Done}$ contains all nodes in $k_{\text{Reachable}}$, a node at depth $k + 1$ has an optimal next-hop path. $N_{16}$ states that once consistent distances are obtained and
Done contains all nodes in \(k_{\text{Reachable}}\), within one step, outgoing channels of a node pair \((u, z)\) at depth \(k + 1\) will not contain any messages for \(z\).

\(N_{13}\) follows from \(N_{15}, N_{16},\) and \(N_{14}\) by closure.

\(N_{15}\) follows from \(D_3, E_1\) and \(M_{12}\) by the implication. (The details are as follows: Consider \((u, z)\) satisfying the left side of \(N_{15}\). Consider \(v\), a next node on a shortest length optimal path from \(u\) to \(z\). From the definition of depth, depth\((v, z)\) = \(k\). From the left side of \(N_{15}\), the outgoing channels of \(v\) do not contain any messages for \(z\). Hence, from \(D_3\), Routevia\(_v\)\((v, z)\) is an optimal path. From Consistent_distances (left side of \(N_{15}\)) and \(E_1\), distances via all other neighbors of \(u\) equal cost of some path. Hence, from \(M_{12}\), \(u\) has an optimal path.) Thus it suffices to prove \(N_{16}\).

\(N_{17}\) states that once Consistent_distances is achieved, and Done contains \(k_{\text{Reachable}}\), the number of messages in the outgoing channels of a node \(u\) at depth \(k + 1\) does not increase.

\(N_{16}\) follows from \(N_{17}\) and \(B_6\) by closure. \(N_{17}\) follows from the unless rule. This completes the proof of the theorem.

End of proof of Theorem 3

5 Algorithm A3

Table 3 specifies the state variables and events of a node in A3. (Refer to the table in the following discussion.) A3 differs from A2 only in the procedure Update&Send.

In A3, the node id's are considered totally ordered. Node \(u\) chooses a neighbor \(v\) as its next-hop for destination \(z\) iff (i) \(v\) is the minimum node in Best_hops\(_u\)(\(z\)), and (ii) \(v\) is the minimum node in Best_hops\(_x\)(\(z\)) for every node \(x\) in the route to \(z\) via \(v\). If there is no such \(v\), then the next-hop is nil. (See definition of Min_best_hop(z) in the table.)

Procedure Update&Send considers a destination \(z\) as affected if (1) distance for \(z\) has changed, or (2) route for \(z\) has changed, or (3) some node \(x\) on Route\(_u\)(\(z\)) is affected. This ensures that if the next-hop changes for a destination \(z\), which is on the route to another destination \(z\), the next-hop for \(z\) also changes.

This way of choosing next-hops and affected destinations ensures that during convergence (when the routes are not stable), the following property \(P\) holds: the next-hop of \(u\) for destination \(z\) is also the next-hop for all intermediate destinations on Route\(_u\)(\(z\)).

Note that in A3, node \(u\) may choose the next-hop for destination \(z\) to be nil, when in fact there is a
neighbor \( v \), and choosing \( v \) as the next-hop to \( z \) satisfies \( P \). Although it may seem that this slows down the convergence, there is a good reason for doing this: if the minimum node in \( \text{Best}_2(z) \), say \( w \), does not satisfy \( P \), then it means that \( u \) has inconsistent distances via \( v \) and \( w \).

**Theorem 4.** \( A3 \) satisfies \( M_1 \) and \( M_2 \).

**Proof of Theorem 4**

Proof of Theorem 4 is identical to proof of Theorem 2 with the following changes. We redefine \( In \) and \( \text{Lowest} \) for \( A3 \) as follows:

\[ \text{In.} \quad \text{Maximal subset of} \ \text{Reachable} \text{ such that } (u, z) \text{ is a member of} \ \text{In} \iff \\
(1) \quad \text{Has} \ \text{optimal}_- \text{path}(u, z), \\
(2) \quad \text{for any message } (x, d, p, cs, rd) \text{ in transit, } \text{Dist}_u(z) \text{ is less than } rd, \\
(3) \quad \text{for any node pair } (w, x) \text{ in} \ \text{Reachable} \text{ not satisfying} \ \text{Has} \ \text{optimal}_- \text{path}(w, x), \\
\text{Dist}_u(z) < \min\{\text{Distvia}_u(v, x) : v \in \text{neighbors}(w)\} \text{ and } \text{Dist}_u(z) < \text{Cost}(w, x). \]

\[ \text{Lowest.} \quad \text{Formally,} \\
= \min(\{\text{Cost}(x, z) : (x, z) \in \text{Out}\}, \\
\{\text{Distvia}_u(v, z) : (u, z) \in \text{Out} \land v \in \text{neighbors}(u)\}, \\
\{rd : (x, d, p, cs, rd) \in \text{Channel}_{uv} \land (u, x), (v, x) \in \text{Reachable}\}). \]

We modify the assertions \( M_{12} \) and \( M_{14} \) as follows:

\( M_{12} \) \quad \text{Invariant} \quad u \neq z \land \text{Dist}_u(z) \neq \infty \Rightarrow \text{Dist}_u(z) = \min\{\text{Distvia}_u(v, z) : v \in \text{neighbors}(u)\} \]

\( M_{14} \) \quad \text{Invariant} \quad \text{Cost}(u, z) \leq \text{Lowest} \land u \neq z \\
\Rightarrow [3(v, z) \in \text{In} : \text{Distvia}_u(v, z) = \text{Cost}(u, z)] \land \\
[\forall(v, z) \in \text{In} : \text{Distvia}_u(v, z) = \text{Cost}(u, z) \land x \in \text{Routevia}_u(v, z) \Rightarrow \text{Distvia}_u(v, x) = \text{Cost}(u, x)]

\( M_{12} \) follows from invariance rule. \( M_{14} \) follows from \( B_2 \) and \( B_3 \) by implication.

Other assertions that hold for \( A2 \) also hold for \( A3 \). Their proofs are identical except \( M_{11} \) which now follows from \( M_{20}, M_{21}, M_{12}, M_{13} \) and \( M_{14} \) by implication where \( M_{20} \) and \( M_{21} \) are as follows:

\( M_{20} \) \quad \text{Invariant} \quad [\exists v \in \text{neighbors}(u) : [\forall x \in \text{Routevia}_u(v, z) : v = \min\text{Best}_2\text{hops}_u(x)]] \Rightarrow \text{Dist}_u(z) \neq \infty \\
\( M_{21} \) \quad \text{Invariant} \quad \text{Lowest} \geq k \land |\text{Diff}_k| = 0 \land \text{Distancevia}_u(v, z) = \text{Cost}(u, z) = k \\
\land x \in \text{Routevia}_u(v, z) \land \text{Distvia}_u(w, x) = \text{Cost}(u, x) \\
\Rightarrow \text{Distancevia}_u(w, z) = \text{Cost}(u, z)

\( M_{20} \) follows from invariance rule. \( M_{21} \) follows from \( B_2 \) and \( B_3 \) by implication. Hence \( M_1 \) and \( M_2 \) hold for
A3.

End of proof of Theorem 4

Theorem 5. Assuming synchronous execution, A3 satisfies N1 and N2.

Proof of Theorem 5

Proof of Theorem 5 is identical to proof of Theorem 3 with \( \text{depth}(u, z) \) being redefined. In A2, \( \text{depth}(u, z) \) stood for minimum length of an optimal path from \( u \) to \( z \). Many such paths can exist and any of them can be chosen by \( u \). In A3, only one of these optimal paths can be chosen by \( u \); i.e. the path \( p = \langle x_0, x_1, \ldots, x_n \rangle \) where \( x_0 = u, x_n = z \) and \( x_1 \) is the minimum-id neighbor of the neighbors on the optimal paths to \( x_i \), for \( i = 1, \ldots, n \). (Note that this may not be the shortest length optimal path.) We redefine \( \text{depth}(u, z) \) to handle this:

\[
\text{depth}(u, z) = \|p\| \text{ such that } p \in \text{AvailablePaths}(u, z) \land \text{PathCost}(p) = \text{Cost}(u, z) \land \\
(\forall x \in p : \text{front}(\text{tail}(p)) = \min\{\text{front}(\text{tail}(q)) : q \in \text{AvailablePaths}(u, x) \land \\
\text{PathCost}(q) = \text{Cost}(u, x)\}).
\]

All assertions that hold for A2 also hold for A3. Their proofs are identical except \( N_{15} \) now follows from \( M_{20}, M_{12} \) (as defined in the proof of Theorem 4), \( E_1 \) and \( D_3 \) by implication. Hence \( N_1 \) and \( N_2 \) hold for A3.

End of proof of Theorem 5

6 Algorithm A4

Table 4 specifies the state variables and events of a node in A4. Each node \( u \) maintains the state variables of A3, except that \( \text{Routevia}_u(v, z) \) and \( \text{Route}_u(z) \) are now auxiliary. Instead, node \( u \) maintains new state variables \( \text{Pfnodevia}_u(v, z) \) and \( \text{Pfnode}_u(z) \). In \( \text{Pfnodevia}_u(v, z) \), node \( u \) maintains an estimate of the prefinal node via neighbor \( v \) (i.e. the last node before \( z \) on the path to \( z \) via \( v \)). \( \text{Pfnodevia}_u(v, z) \) is equal to \( \text{nil} \) if node \( u \) believes \( z \) cannot be reached via \( v \). State variable \( \text{Pfnode}_u(z) \) indicates the prefinal node via node \( u \)'s next-hop.

The messages of A4 are like the message of A3, except that they now contain prefinal node information, and the route information is auxiliary (i.e. not implemented).

The events of algorithm A4 are like those of algorithm A3, with the following twist: each node in A4 uses its prefinal nodes to construct prefinal-routes, which take the place of the routes in A3. Node
$u$ constructs its prefinal-route via neighbor $v$ for destination $z$, referred to as $Pfroute_u(v, z)$, as follows: Start with a sequence $\langle z \rangle$; add to the left of this sequence the prefinal node via $v$ for the leftmost element of the sequence, until either (1) node $u$ is added, or (2) the prefinal node is nil, or (3) a loop is established. We use $Pfroute_u(z)$ to refer to the prefinal-route for destination $z$ via the next-hop. (Formal definitions of functions $Pfroute_u(v, z)$ and $Pfroute_u(z)$ are in the table).

**Theorem 6.**

(a) $A4$ satisfies $M_1$ and $M_2$

(b) Assuming synchronous execution, $A4$ satisfies $N_1$ and $N_2$.

**Proof of Theorem 6**

Because the variables of $A4$ (both auxiliary and non-auxiliary) are a superset of the non-auxiliary variables of $A3$ and their domains are the same, there is a natural (projection) mapping from the states of $A4$ to the states of $A3$. For any state $s$ of $A4$, let $s'$ denote the corresponding state of $A3$. It is obvious that event $e$ of $A4$ is enabled in any state $s$ iff the corresponding event $e$ of $A3$ is enabled in $s'$. We next show that event $e$ of $A4$ updates the variables of $A3$ in the same way as the corresponding event $e$ of $A3$, more precisely, if event $e$ of $A4$ has a transition $(s, t)$, then the corresponding event $e$ of $A3$ has a transition $(s', t')$. For this, it is sufficient to establish that the prefinal-routes of $A4$ simulate accurately the routes of $A3$. This is specified by the following assertion:

\[
\text{(R1) Invariant} \quad \left(\langle Routevia_u(v, z) = Pfroutevia_u(v, z) \rangle \land \langle Routevia_u(v, z) = \langle \rangle \land Pfroutevia_u(v, z) = \langle z \rangle \rangle \land \langle Routevia_u(z) = Pfroutevia_u(z) \rangle \lor \langle Routevia_u(z) = \langle \rangle \land Pfroutevia_u(z) = \langle z \rangle \rangle \land \langle (z, d, pf_n, p) = Channel_{uv}[j] \Rightarrow \langle p = PfMroute_{uv}(j, z) \rangle \lor \langle p = \langle \rangle \land PfMroute_{uv}(j, z) = \langle z \rangle \rangle \right)
\]

where $Channel_{uv}[j]$ denotes the $j$-th message in $Channel_{uv}$ ($Channel_{uv}[0]$ is the front) and $PfMroute_{uv}(j, z)$ is defined as follows:

$PfMroute_{uv}(j, z)$. Sequence of nodes. $\langle s_0, \ldots, s_n \rangle$ where

(a) $s_n = z$,

(b) for all $i \in [0..n-1]$: $s_i = \begin{cases} Pfnodevia_u(u, s_{i+1}) & \text{if for all } k \leq j, Channel_{uv}[k] \text{ does not contain } (s_{i+1}, d, pf_n, p) \\ pf_n & \text{if for largest } k \leq j, Channel_{uv}[k] \text{ contains } (s_{i+1}, d, pf_n, p) \end{cases}$

(c) for all $i \in [1..n-1]$: $s_i \notin \{s_{i+1}, \ldots, s_n\}$, and

(d) $s_0 = u \lor Pfnodevia_u(s_0) = nil \lor s_0 \in \{s_1, \ldots, s_n\}$.

$R_1$ states that the prefinal-routes and the routes (which are auxiliary) agree. $R_1$ follows from invariance
rule.

Given $R_1$, if event $e$ of $A_4$ has transition $(s, t)$, then the corresponding event $e$ of $A_3$ has transition $(s', t')$. We have already established that $e$ of $A_4$ is enabled whenever $e$ of $A_3$ is enabled. We also have that the initial condition of $A_4$ imply the initial condition of $A_3$. Thus, $A_4$ is a strongly well-formed refinement of $A_3$; that is, $A_4$ satisfies any safety or progress properties of $A_3$. This and Theorem 4 imply Theorem 5.

End of proof of Theorem 6

7 Concluding Remarks

The algorithms analyzed in this paper are representative of various internetworking distance-vector routing protocols. Distance-vector routing algorithms are difficult to understand. Most of their analyses in the literature is operational. In the course of our work, we discovered that they are often incomplete or inaccurate; for example, reference [17] considers only one or two link failures rather than an arbitrary succession of topology changes, to prove the properties of their algorithm; the routing table update procedure in [3] is inaccurate; the example in [3] to illustrate $O(N)$ convergence is wrong, etc. A stepwise assertional design, such as the one presented here, is effective at making it easier to understand these algorithms.

In our opinion, the major drawback of our stepwise design is that we could not obtain a refinement result for algorithm $A_2$ and $A_3$ similar to the result for algorithm $A_4$. Instead, we had to check that the proof that $A_1$ eventually achieves optimal paths also holds for $A_2$ and $A_3$, and that the proof that $A_2$ achieves optimal paths in $N + H$ steps also holds for $A_3$.

References


State variables and initial conditions of node $u$:

- $\text{Linkcost}_u(v) : I^+ \cup \{\infty\}$. Initially $\infty$. Cost of the link $(u, v)$.
- $\text{Distvia}_u(v, z) : I^+ \cup \{\infty\}$. Initially $\infty$. Distance to destination $z$ via neighbor $v$.
- $\text{Nhops}_u(z) : \text{neighbors}(u) \cup \{\text{nil}\}$. Initially $\text{nil}$. Next-hop for destination $z$.
- $\text{Dist}_u(z) : I^+ \cup \{0, \infty\}$. Initially $\infty$ for $u \neq z$, and $0$ for $u = z$. Distance to destination $z$ via next-hop.

Events of node $u$:

- **Receive**$_u(v, d_{\text{vector}})$
  - action: $\text{Update} \& \text{Send}_u(v, \{(z, d + \text{Linkcost}_u(v)) : (z, d) \in d_{\text{vector}}\})$  
    \{ $d$ can be $\infty$ \}

- **LinkCostChange**$_u(v, \text{newcost})$
  - enabled: $\text{Linkcost}_u(v) < \infty$
  - action: $c := \text{newcost} - \text{Linkcost}_u(v)$; $\text{Linkcost}_u(v) := \text{newcost}$;
    $\text{Update} \& \text{Send}_u(v, \{(z, \text{Distvia}_u(v, z) + c) : \forall z \in \text{NODES}\})$

- **LinkFailure**$_u(v)$
  - enabled: $\text{Linkcost}_u(v) < \infty$
  - action: $\text{Channel}_u := \text{nil}$; $\text{Linkcost}_u(v) := \infty$;
    $\text{Update} \& \text{Send}_u(v, \{(z, \infty) : \forall z \in \text{NODES}\})$

- **LinkRecovery**$_u(v, \text{newcost})$
  - enabled: $\text{Linkcost}_u(v) = \infty$
  - action: $\text{Linkcost}_u(v) := \text{newcost}$;
    $\text{Update} \& \text{Send}_u(v, \{(v, \text{newcost})\})$;
    $\text{Send}(u, \{(z, \text{Dist}_u(z)) : \forall z \in \text{NODES}\})$ to $v$

- **Update**$\&$**Send**$_u(v, d_{\text{vector}})$
  - local variable affectedsinks initially $\{\}$
  - for all $(z, d) \in d_{\text{vector}}$ do
    - $\text{Distvia}_u(v, z) := d$; \{ Note that $d$ can be $\infty$ \}
    - if $\text{Nhops}_u(z) \neq v \wedge \text{Distvia}_u(v, z) < \text{Dist}_u(z)$
      - $\text{Nhops}_u(z) := v$; $\text{Distvia}_u(v, z) := \text{Dist}_u(z)$
      - if $\text{BestNhops}_u(z) \neq \{\}$ then
        - for some $k \in \text{BestNhops}_u(z)$ do
          - $\text{Nhops}_u(z) := k$; $\text{Dist}_u(z) := \text{Distvia}_u(k, z)$
        - else
          - $\text{Nhops}_u(z) := \text{nil}$; $\text{Dist}_u(z) := \infty$
      endif;
    - affectedsinks := affectedsinks $\cup \{z\}$
  endif
  for all $w$ such that $(u, w) \in \text{UPLINKS}$ do
    $\text{Send}(u, \{(z, \text{Dist}_u(z)) : \forall z \in \text{affectedsinks}\})$ to $w$
  endfor

where $\text{BestNhops}_u(z)$ is a function that returns the following subset of $\text{neighbors}(u)$:

\[
\{v : \text{Distvia}_u(v, z) \neq \infty \wedge \text{Distvia}_u(v, z) = \min\{\text{Distvia}_u(w, z) : w \in \text{neighbors}(u)\}\}
\]
State variables and initial conditions of node $u$:

- $LinkCost_u(v), Nhop_u(z), Dist_u(z), Distvia_u(v, z)$. As in A1.
- $Routevia_u(v, z)$. sequence of nodes. Initially $\emptyset$. Path from $u$ to $z$ via $v$.
- $Costseq_u(v, z)$, sequence of $I^+ \cup \{\infty\}$. Auxiliary. Initially $\emptyset$. Sequence of link costs on $Routevia_u(v, z)$.
- $Route_u(z)$. sequence of nodes. Initially $\emptyset$ for $u \neq z$, $\langle u \rangle$ for $u = z$. Path from $u$ to $z$.
- $Costseq_u(z)$, sequence of $I^+ \cup \{\infty\}$. Auxiliary. Initially $\emptyset$ for $u \neq z$, $\langle 0 \rangle$ for $u = z$. Sequence of link costs on $Route_u(z)$.

Events of node $u$:

- $Receive_u(v, dvector)$
  - action:
    - local variable $dvector2$: initially $\emptyset$.
    - $dvector2 := \{(z, d + LinkCost_u(v), \langle u \rangle@p, \langle LinkCost_u(v) \rangle@cs) : (z, d, p, cs, rd) \in dvector \land d \neq \infty\}$
    - $U\_UpdateSend_u(v, dvector2)$

- $LinkCostChange_u(v, newcost) \quad \{newcost \neq \infty\}$
  - enabled: $LinkCost_u(v) < \infty$
  - action:
    - local variable $dvector$: initially $\emptyset$.
    - $c := newcost - LinkCost_u(v)$; $LinkCost_u(v) := newcost$;
    - $dvector := \{(z, Distvia_u(v, z) + c, Routevia_u(v, z), \langle newcost \rangle @ tail(Costseqvia_u(v, z))) : \forall z \in NODES\}$
    - $U\_UpdateSend_u(v, dvector)$

- $LinkFailure_u(v)$
  - enabled: $LinkCost_u(v) < \infty$
  - action:
    - Channel_u := $\emptyset$; $LinkCost_u(v) := \infty$;
    - $U\_UpdateSend_u(v, \{\langle v, newcost, \langle u, v, \langle LinkCost_u(v), 0 \rangle \rangle \})$;
    - $Send(u, \{(z, Dist_u(z), Route_u(z), Costseq_u(z), Dist_u(z)) : \forall z \in NODES \land v \notin Route_u(z)\}$
    - $\cup \{(\langle z, \infty, \langle z \rangle, \langle \rangle, \langle \rangle) : \forall z \in NODES \land v \in Route_u(z)\}$

- $UpdateSend_u(v, dvector)$
  - local variable $affectedsink$: initially $\emptyset$.
  - for all $(z, d, p, cs) \in dvector$ do
    - Note that $d$ can be $\infty$
    - $Distvia_u(v, z) := d$; $Routevia_u(v, z) := p$; $Costseqvia_u(v, z) := cs$;
    - if $(Nhop_u(z) \neq v \land Distvia_u(v, z) < Distvia_u(z)$)
      - $\lor (Nhop_u(z) = v \land (Distvia_u(v, z) \neq Distvia_u(z) \lor \forall z \in Routevia_u(v, z) \neq Routevia_u(z)$)
      - if $Besthops_u(z) \neq \emptyset$ then
        - for some $k \in Besthops_u(z)$ do
          - $Nhop_u(z) := k$; $Distvia_u(k, z) := Distvia_u(k, z)$;
          - $Routevia_u(z) := Routevia_u(k, z)$; $Costseqvia_u(z) := Costseqvia_u(k, z)$
        - else
          - $Nhop_u(z) := \text{nil}$; $Distvia_u(z) := \infty$; $Routevia_u(z) := \emptyset$; $Costseqvia_u(z) := \emptyset$
      - endif;
      - $affectedsink := affectedsink \cup \{z\}$
    - endif
  - for all $w$ such that $(u, w) \in UPLINKS$ do
    - local variable $dvector$: initially $\emptyset$.
    - $dvector := \{(\langle z, \infty, \langle \rangle, \langle \rangle, \langle \rangle, \langle \rangle) : (z, \langle \rangle, \langle \rangle, \langle \rangle, \langle \rangle) \in \forall \langle \rangle \in Route_u(z) \land \exists \langle \rangle \in z \in affectedsink\}$
    - $Send(u, dvector)$ to $w$;

where $Besthops_u(z)$ is as defined in A1 (Table 1).
Table 3: Algorithm A3.

State variables and initial conditions of node u:

- \(\text{Linkcost}_u(v)\)
- \(\text{Nhopt}_u(z)\)
- \(\text{Dist}_u(z)\)
- \(\text{Distvia}_u(v,z)\)
- \(\text{Route}_u(z)\)
- \(\text{Routevia}_u(v,z)\)
- \(\text{Costseq}_u(z)\)
- \(\text{Costseqvia}_u(v,z)\)

- As in A2.

Events of node u:

- \(\text{Receive}_u, \text{LinkCostChange}_u, \text{LinkFailure}_u, \text{LinkRecovery}_u\).

- As in A2.

- \(\text{Update}\&\text{Send}_u(v,d_{vector})\)

  - local variable \(\text{affectedsink}\)s: initially \(\{\}\);
    
    - for all \((z, d, p, cs) \in d_{vector}\) do 
      
      \(\text{Distvia}_u(v,z) := d; \text{Routevia}_u(v,z) := p; \text{Costseqvia}_u(v,z) := cs;\)
      
      \(\text{if } (\text{Nhopt}_u(z) \neq v \land \text{Distvia}_u(v,z) < \text{Dist}_u(z)) \lor (\text{Nhopt}_u(z) = v \land (\text{Distvia}_u(v,z) \neq \text{Dist}_u(z) \lor \text{Routevia}_u(v,z) \neq \text{Route}_u(z))) \text{ then}\)

      \(\text{affectedsink} := \text{affectedsink} \cup \{z\}\)

    - end if

    - for all \(z \in \text{NODES} - \text{affectedsink}\) do 
      
      \(\text{if } \{\exists x : x \in \text{Route}_u(z) \land x \in \text{affectedsink}\} \text{ then}\)

      \(\text{affectedsink} := \text{affectedsink} \cup \{z\}\)

    - end if

  - end for

  - for all \(z \in \text{affectedsink}\) do 
    
    - if \(\text{Min\textunderscore best\textunderscore hop}_u(z) \neq \{\}\) then
      
      \(\text{for some } k \in \text{Min\textunderscore best\textunderscore hops}_u(z) \text{ do }\)

      \(\text{Nhopt}_u(z) := k; \text{Dist}_u(z) := \text{Distvia}_u(k,z); \text{Route}_u(z) := \text{Routevia}_u(k,z); \text{Costseq}_u(z) := \text{Costseqvia}_u(k,z)\)

    else

      \(\text{Nhopt}_u(z) := \text{nil}; \text{Dist}_u(z) := \infty; \text{Route}_u(z) := \{\}; \text{Costseq}_u(z) := \{\}\)

    end if

  - end for

  - for all \(w\) such that \((u, w) \in \text{UPLINKS}\) do 
    
    - local variable \(d_{vector}\) : initially \(\{\}\);
      
      \(d_{vector} := \{(z, \infty, \{\}, \{\}, \text{Dist}_u(z)) : w \in \text{Route}_u(z) \land z \in \text{affectedsink}\}\)

      \(\cup \{(z, \text{Dist}_u(z), \text{Route}_u(z), \text{Costseq}_u(z), \text{Dist}_u(z)) : w \notin \text{Route}_u(z) \land z \in \text{affectedsink}\}\); 

    - Send\((u, d_{vector})\) to \(w\);

  - end for

where the function \(\text{Min\textunderscore best\textunderscore hop}_u(z)\) is now defined as follows:

- \(\{v : \{\forall x \in \text{Routevia}_u(v,z) : v = \text{Min\textunderscore best\textunderscore hops}_u(x)\}\}\)

where the function \(\text{Best\textunderscore hops}_u(x)\) is as defined in A1.
Table 4: Algorithm A4.

State variables and initial conditions of node $u$:
- $\text{Linkcost}_u(v)$, $\text{Nhops}_u(z)$, $\text{Dist}_u(z)$, $\text{Distvia}_u(v, z)$. As in A3.
- $\text{Route}_u(z)$, $\text{Routevia}_u(v, z)$. Auxiliary. As in A3.
- $\text{Pfnode}_u(v, z)$: neighbors$(z) \cup \{\text{nil}\}$. Initially nil. Prefinal node on the path from $u$ to $z$ via $v$.
- $\text{Pfnode}_u(z)$: neighbors$(z) \cup \{\text{nil}\}$. Initially nil. Prefinal node on the path from $u$ to $z$.

Functions:
- $\text{Pfroute}_u(z)$: sequence of nodes $\langle s_0, \ldots, s_n \rangle$ where $s_n = z$, for all $i \in [0..n-1]$: $s_i = \text{Pfnode}_u(s_{i+1})$, for all $i \in [1..n-1]$: $s_i \notin \{s_{i+1}, \ldots, s_n\}$, and $s_0 = u \lor \text{Pfnode}_u(s_0) = \text{nil} \lor \infty \in \{s_1, \ldots, s_n\}$.
- $\text{Pfroutevia}_u(v, z)$: Defined like $\text{Pfroute}_u(z)$ except $\text{Pfnode}_u(x)$ is replaced by $\text{Pfnodevia}_u(v, x)$.

Events of node $u$:
- $\text{Receive}_u(v, \text{d_vector})$
  - action: local variable $\text{d_vector}2$ : initially $\{\}$;
  - $\text{d_vector2} := \{(z, \infty, \text{nil}, \langle \rangle): (z, d, \text{pf}, p) \in \text{d_vector} \land d = \infty\}$
  - $\text{U} \cup \{(z, d + \text{Linkcost}_u(v), u, \langle \rangle): (z, d, \text{pf}, p) \in \text{d_vector} \land z = v \land d \neq \infty\}$
  - $\text{U} \cup \{(z, d + \text{Linkcost}_u(v), \text{pf}, p, u) \in \text{d_vector} \land z \neq v \land d \neq \infty\}$
  - $\text{Update}&\text{Send}_u(v, \text{d_vector}2)$

- $\text{LinkCostChange}_u(v, \text{newcost})$ \{newcost $\neq \infty$\}
  - enabled: $\text{Linkcost}_u(v) < \infty$
  - action: local variable $\text{d_vector}$ : initially $\{\}$; $c$;
  - $c := \text{newcost} - \text{Linkcost}_u(v)$; $\text{Linkcost}_u(v) := \text{newcost}$;
  - $\text{d_vector} := \{(z, \text{Distvia}_u(v, z) + c, \text{Pfnodevia}_u(v, z), \text{Routevia}_u(v, z)): \forall z \in \text{NODES}\}$;
  - $\text{Update}&\text{Send}_u(v, \text{d_vector})$

- $\text{LinkFailure}_u(v)$
  - enabled: $\text{Linkcost}_u(v) < \infty$
  - action: $\text{Channel}_u(v) := \{\}; \text{Linkcost}_u(v) := \text{newcost}$;
  - $\text{Update}&\text{Send}_u(v, \{(z, \infty, \text{nil}, \langle \rangle): \forall z \in \text{NODES}\})$

- $\text{LinkRecovery}_u(v, \text{newcost})$ \{newcost $\neq \infty$\}
  - enabled: $\text{Linkcost}_u(v) = \infty$
  - action: $\text{Linkcost}_u(v) := \text{newcost}$;
  - $\text{Update}&\text{Send}_u(v, \{(v, \text{newcost}, u, \langle u, v \rangle)\})$
  - $\text{Send}_u(v, \{(z, \infty, \text{nil}, \langle \rangle): v \in \text{Pfroute}_u(z) \land z \in \text{affectedsinks}\})$
  - $\text{U} \cup \{(z, \text{Dist}_u(z), \text{Pfnode}_u(z), \text{Route}_u(z)): v \notin \text{Pfroute}_u(z) \land z \in \text{affectedsinks}\}$ to $v$
Table 4 (cont.): Algorithm A4.

$\text{Update:Send}_u(v,d\text{-vector})$

local variable $\text{affectedsinks}$: initially $\{\}$;

for all $(z,d,pfn,p) \in d\text{-vector}$ do  
  {Note that $d$ can be $\infty$}
  \hspace{1em} $\text{Dist}_u(v,z):=d$; $\text{Pfnode}_u(v,z):=pfn$; $\text{Route}_u(v,z):=p$;
  \hspace{1em} if ($\text{Nhop}_u(z) \neq v \land \text{Dist}_u(v,z) < \text{Dist}_u(z)$)
  \hspace{3em} $\lor$ ($\text{Nhop}_u(z) = v \land (\text{Dist}_u(v,z) \neq \text{Dist}_u(z) \lor \text{Pfroute}_u(z) \neq \text{Pfroute}_u(v,z))$) then
  \hspace{5em} $\text{affectedsinks}:= \text{affectedsinks} \cup \{z\}$
  \hspace{1em} endif
  \hspace{1em} for all $z \in \text{NODES-affectedsinks}$ do
  \hspace{3em} if [$\exists k : k \in \text{Pfroute}_u(z) \land k \in \text{affectedsinks}$] then
  \hspace{5em} $\text{affectedsinks}:= \text{affectedsinks} \cup \{z\}$
  \hspace{1em} endif
  \hspace{1em} for all $z \in \text{affectedsinks}$ do
  \hspace{3em} if $\text{Min}\_\text{Best\_hop}_u(z) \neq \{\}$ then
  \hspace{5em} for some $k \in \text{Min}\_\text{Best\_hop}_u(z)$ do
    \hspace{7em} $\text{Nhop}_u(z) := k$; $\text{Dist}_u(z) := \text{Dist}_u(k,z)$;
    \hspace{7em} $\text{Pfnode}_u(z) := \text{Pfroute}_u(k,z)$; $\text{Route}_u(z) := \text{Route}_u(k,z)$;
  \hspace{5em} else
    \hspace{7em} $\text{Nhop}_u(z) := \text{nil}$; $\text{Dist}_u(z) := \infty$; $\text{Pfnode}_u(z) := \text{nil}$; $\text{Route}_u(z) := \{\}$;
  \hspace{5em} endif
  \hspace{3em} for all $w$ such that $(u,w) \in \text{UPLINKS}$ do
  \hspace{5em} local variable $d\text{-vector}$: initially $\{\}$;
  \hspace{6em} $d\text{-vector} := \{(z,\infty,\text{nil},\{\}) : w \in \text{Pfroute}_u(z) \land z \in \text{affectedsinks}\} \cup \{(z,\text{Dist}_u(z),\text{Pfnode}_u(z),\text{Route}_u(z)) : w \notin \text{Pfroute}_u(z) \land z \in \text{affectedsinks}\}$;
  \hspace{5em} Send$(u,d\text{-vector})$ to $w$;
  \hspace{3em} endif
\hspace{1em} endif
\hspace{1em} endif

where the function $\text{Min}\_\text{Best\_hop}_u(z)$ is now defined as follows:

$\{v : \forall x \in \text{Pfroute}_u(v,z) : v = \text{min}\_\text{Best\_hop}_u(x)\}$

where the function $\text{Best\_hop}_u(x)$ is as defined in A1.