# Routing metrics design for multihop wireless networks

**Yaling Yang**
*yyang8@vt.edu*
*Virginia Polytechnic Institute and State University*

**Jun Wang**
*junwan@microsoft.com*
*Microsoft Corporation*

***Abstract*— The design of a routing protocol must be based on the characteristics of its target networks. The diversity of wireless networks motivate the design of different routing metrics, capturing different aspects of wireless communications. The design of routing metrics, however, is not arbitrary since it has a great impact on the proper operation of routing protocols. Combining a wrong type of routing metrics with a routing protocol may result in routing loops and suboptimal paths. In this paper, we thoroughly study the relationship between routing metrics and routing protocols. Our work provides several guidelines for designing routing metrcs and identify the specific properties that a routing metric must have in order to be combined with certain type of routing protocols.**

## I. Introduction

The main goal of any routing protocol is to support effective communications. To achieve this goal, the design of a routing protocol must be based on the characteristics of its target networks. This demand for network specific design is especially important for wireless networks due to the severely limited resources and highly dynamic nature of wireless communications. For example, the severe energy constraints of sensor networks demand the design of energy efficient routing. The mobility of nodes in ad hoc networks demands routing protocols that can converge rapidly. The heavy traffic load in mesh networks requires load-balanced routing schemes.

To satisfy these different requirements in different networks, existing literature focuses on designing different routing metrics for different wireless networks. While these network specific routing metrics can capture the characteristics of their target networks, there is a lack of knowledge about the impact of these routing metric designs on the operations of routing protocols. In fact, we cannot arbitrarily exchange one routing metric to another without considering the routing protocols used in the network. If one combines the wrong type of routing protocol with a routing metric, the routing protocol may fail to find an optimal path or even create routing loops. The focus of our research, hence, is to provide a systematic analysis of the relationship between routing metrics and routing protocols. Our research identifies the basic properties that a routing metric must have in order to work with different proactive or reactive routing protocols in wireless networks. The results of our research provides an important guideline for selecting the appropriate combinations of routing metrics and routing protocols.

The reminder of the paper is organized as follows. Section II briefly reviews different types of routing protocols that are widely used in wireless networks and presents three requirements for these routing protocols. Section III introduces the properties of routing metrics that are critical for routing protocols to satisfy their three requirements. Sections IV, V and VI analyze the different required properties of routing metrics when the routing metrics are combined with different type of routing protocols. Section VII presents some examples for using our analytical results to design routing protocols. Section VIII concludes our work and discusses future directions.

## II. Routing Protocols and Their Requirements

Different routing metrics are designed to capture different characteristics of a path. For example, some routing metrics capture the stability of a path, some focus on energy consumption, and some are more concerned about the bandwidth of a path. Depending on the design, many routing metrics cannot be combined with an arbitrary routing protocol. If they are combined with the wrong type of routing protocols, these routing protocols may not work properly. In this section, we introduce the necessary concepts for describing the proper operations of different routing protocols. We first briefly review the different types of routing protocols that are used in wireless networks. Then, we establish a mathematical model of wireless networks. Finally, using the mathematical model, we formally define the three requirements that a properly operated routing protocol must satisfy.

### A. Types of Routing Protocols

A routing protocol is composed of two components: a path calculation algorithm and a packet forwarding scheme. Different combinations of path calculation algorithms and packet forwarding schemes result in different routing protocols. In this section, we review the most commonly used path calculation algorithms and packet forwarding schemes in wireless networks. By classifying the routing protocols based on their path calculation algorithms and packet forwarding schemes, we can examine the design guidelines of routing metrics for each class of routing protocols.

*1) Path Calculation Algorithms:* Different path calculation algorithms have different costs and are appropriate for different networks. In this paper, we exam three path calculation algorithms: flooding-based route discovery, Dijkstra's algorithm and the Bellman-Ford algorithm, all of which are widely used in wireless routing protocols.

In flooding-based route discovery, to search for a path to a destination, a source node does not need any preexisting knowledge about the network topology. The source node simply floods a route discovery message through the entire network until this message reaches the destination. During the flooding period, multiple paths are explored by copies of the route discovery message simultaneously and a single path among all the searched paths is selected as the path to the destination. Flooding-based route discovery does not require periodic message exchanges. It can cope with highly dynamic wireless networks such as mobile ad hoc networks. On the other hand, flooding-based route discovery imposes high and bursty control message overhead on the network and hence is only appropriate for networks that are lightly loaded.

In Dijkstra's algorithm or the Bellman-Ford algorithm, to calculate paths to other nodes, a node must have certain level of knowledge about the network topology. Such knowledge is collected through periodic message exchanges among neighboring nodes. For static wireless networks, such as sensor or mesh networks, this type of path calculation algorithms are often preferred due to its low control message overhead compared with flooding-based route discovery.

*2) Packet Forwarding Schemes:* In wireless networks, two packet forwarding schemes, source routing and hop-by-hop routing, are often used in different routing protocols. In source routing, a source node put the entire path of a flow in its packet headers. Intermediate nodes forward packets based on the path information carried in packet headers. In hop-by-hop routing, a source node only puts the destination addresses in its packet headers. An intermediate node maintains a routing table that stores the next hops to reach other nodes. Using this routing table, the intermediate node forwards packets based on their destination addresses.

The majority of existing wireless routing protocols use some combinations of the above three path calculation algorithms and two packet forwarding schemes. For example, AODV [1] uses the combination of flooding-based route discovery and hop-by-hop routing. DSR [2] and MCR [3] use flooding-based route discovery coupled with source routing. LQSR [4] combines Dijkstra's algorithm with source routing. HSR [5], OLSR [6] and HSLS [7] adopt Dijkstra's algorithm and hop-by-hop routing. DSDV [8] combines the Bellman-Ford algorithm with hop-by-hop routing. Our study of the relationship between routing metric design and the different combinations of path calculation algorithms and packet forwarding schemes provides important design guidelines for a wide range of routing protocols in wireless networks.

### B. System Model

Before we start the formal discussion of the requirements for routing protocols, in this section, we define a network model and the mathematical representation of routing metrics, called *path weight structures*. Using these formal definitions, we can then proceed to discuss the requirements for routing protocols in Section II-C.

*1) Network Model:* Formally, a wireless network is defined as a strongly connected directed graph $G(V, E)$, where $V$ is the set of nodes and $E$ is the set of edges representing wireless links between neighboring nodes. An edge from node $x$ to $y$ is represented as $(x, y)$.

A path $p$ from $v_1$ to $v_n$ is denoted as $p(v_1, v_n)$ ($p_{v_1,v_n}$ for short) and $p(v_1, v_n) = \langle v_1, v_2, \cdots, v_n \rangle$, where $v_i, 1 \leq i \leq n$ are nodes on path $p(v_1, v_n)$. A path $p$ is simple if all nodes from $v_1$ to $v_n$ are distinct. If $v_1$ and $v_n$ are the same node, $p(v_1, v_n)$ forms a loop. In the rest of this paper, for simplicity, we use *path* to actually mean *simple path*, unless otherwise mentioned. Given a path $p$ and an on-path node $v$, we use $\pi(p, v)$ and $\varpi(p, v)$ to denote the predecessor and successor (or next-hop) of $v$ on $p$, respectively. If $v$ is the endpoint of $p$, then $\varpi(p, v) = \text{NIL}$; and, if $v$ is the source of $p$, then $\pi(p, v) = \text{NIL}$. In a hop-by-hop network, $\varpi(p, v)$ is essential for each node $v \in V$ if the path $p$ is chosen for some destination. This is because the value of $\varpi(p, v)$ represents the next hop that $v$ stores in its routing table for forwarding packets to the destination. The path calculation algorithm of a routing protocol $\mathcal{R}$ is represented by a function $\mathcal{R}(s, t)$ that returns a path from source node $s$ to destination node $d$.

*2) Path Weight and Its Related Algebra:* The goal of any routing protocol is not only to find a path between a pair of nodes, but also to select the best path among all the possible paths between this pair of nodes. The path selection criteria is usually based on weights of paths, which are real numbers assigned to paths based on the design of routing metrics. Hence, the design of routing metrics is directly connected to the performance of routing protocols. In this section, we present the formal definition of routing metrics as follows.

The design of a routing metric can be formally defined as an algebra on top of a quadruplet $(S, \oplus, w, \preceq)$, where $S$ is the set of link weights, $w$ is a function that maps a path or a link to a weight, $\preceq$ is an order relation and $\oplus$ is the path concatenation operation. We call the quadruplet $(S, \oplus, w, \preceq)$ a *path weight structure* and it is the mathematical representation of a routing metric. The expression $p \oplus q$ denotes the concatenation of two paths $p$ and $q$. The relation $\preceq$ provides a total order of weights, where $w(a) \preceq w(b)$ means "$a$ is lighter (better) than or equal to $b$". More strictly, $w(a) \prec w(b)$ means "$a$ is lighter than $b$". By using different definitions of $w()$ and $\preceq$, a path weight structure can capture different path characteristics, such as delay, bandwidth, hop count, stability and energy consumption. Some examples of path weight structures can be found in Section VII. The goal of any routing protocol, hence, is to route packets along the lightest path between any pair of nodes using the appropriate design of path weight structure.

### C. Requirements for Routing Protocols

Using the formal definitions of path weights and network model introduced in the previous section, the three requirements that a correctly operated routing protocols must satisfy are described in this section. These three requirements are consistency, optimality and loop-freeness. Whether a routing

protocol can satisfy these requirements depends on the properties of the path weight structure used in the routing protocol.

*Definition 1:* **(Consistency)** A routing protocol $\mathcal{R}$ is consistent if for any source $s \in V$, any destination $t \in V$ and any on-path node $v$ of $\mathcal{R}(s,t)$, $v$ forwards packets from $s$ to $v$ to $\varpi(\mathcal{R}(s,t),v)$. Conveniently, a path is consistent if it is generated by a consistent routing protocol.

Simply speaking, for a consistent path, the decisions of packet forwarding made by all nodes along the path should be consistent with each other. If node $v_1$ makes a decision that the traffic to $v_n$ should follow the path $p(v_1,v_n) = \langle v_1, v_2, \cdots, v_n\rangle$, then all the nodes along $p$ should make the same decisions; i.e., $v_2$ should make a decision that traffic to $v_n$ should follow the path $p(v_2,v_n) = \langle v_2, v_3, \cdots, v_n\rangle$, and the same for $v_3$, $v_4$, $\cdots$, and $v_{n-1}$. Note that according to our definition above, all sub-paths of a consistent path are consistent too.

For source routing, since relaying nodes only forward packets according to the paths carried in packet headers, the consistency of routing is automatically satisfied. However, for hop-by-hop routing, since every node only forwards packets based on destination addresses, the best path visioned by the source node of a packet may be different from the ones visioned by the relaying nodes of the packet, creating inconsistency in packet routing. To ensure consistency for hop-by-hop routing, we must have $\text{SUBPATH}(\mathcal{R}(s,t),v,t) = \mathcal{R}(v,t)$ for any on-path node $v$ of $\mathcal{R}(v,t)$, where $\text{SUBPATH}(p,u,v)$ denotes the sub-path of path $p$ between on-path nodes $u$ and $v$.

Consistency is an important requirement for routing protocols since it ensures easy management of routing protocols. Consistency is also related to the other two requirements, optimality and loop-freeness, defined as follows.

*Definition 2:* **(Optimality)** Given a path weight structure $(S,\oplus,w,\preceq)$, a routing protocol $\mathcal{R}$ is said to be optimal if $\mathcal{R}$ can route packets along the lightest path between every pair of nodes in any connected network $G(V,E)$. This means that for any two distinct nodes $s,t \in V$ , $\mathcal{R}(s,t) \preceq p_{s,t}$, where $p_{s,t}$ is any non-empty path between $s$ and $t$.

*Definition 3:* **(Loop-freeness)** A routing protocol $\mathcal{R}$ is said to be loop-free if given any connected network $G(V,E)$, when the routing protocol converges, $\mathcal{R}$ does not forward any packet along a circle.

Optimality is an important requirement to ensure the performance of routing protocols since routing metrics are designed to capture the demands of different wireless networks. If a routing protocol cannot route packets along the lightest paths, it compromises the design goal of its routing metric and may fail to meet the demand of the network.

Inconsistent routing can cause violations of optimality. For example, a source node may select the lightest path to route its packets, but a relaying node may route these packets through another path that is not the lightest path due to inconsistency.

Loop-freeness is the most important requirement for a routing protocol. While sub-optimal and inconsistent routing may be tolerated in some networks, a routing protocol that creates routing loops is not usable.

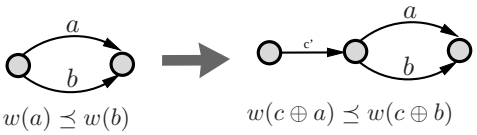


Fig. 1. Example of left-isotonicity

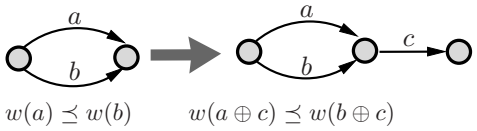


Fig. 2. Example of right-isotonicity

For source routing, loop-freeness can be easily ensured since the source node can eliminate the circles in a path. For hop-by-hop routing, as long as the path calculation algorithm can find a valid path between any pair of source and destination nodes, consistency is sufficient to ensure loop-freeness. This is because in consistent routing, a packet is always forwarded along the path selected by its source node. Hence, as long as the path calculated at the source is a valid path, a packet will never encounter any forwarding circle in routing. The reverse, however, may not hold. An inconsistent routing scheme may or may not create routing loops.

## III. Properties of Weight Structure

Whether a routing protocol can satisfy the optimality, consistency and loop-freeness requirements depends on different types of isotonicity and monotonicity properties of the weight structure. In this section, we introduce these properties and analyze the relationship between isotonicity and lightest paths. In Sections IV, V and VI, we will identify which of these properties are needed for different routing protocols and summarize the results in Table I.

### A. Definitions of Isotonicity and Monotonicity

*Definition 4:* **(Left-isotonicity)** The quadruplet $(S,\oplus,w,\preceq)$ is left-isotonic if $w(a) \preceq w(b)$ implies $w(c \oplus a) \preceq w(c \oplus b)$, for all $a,b,c \in S$. Similarly, $(S,\oplus,w,\preceq)$ is strictly left-isotonic if $w(a) \prec w(b)$ implies $w(c \oplus a) \prec w(c \oplus b)$, for all $a,b,c \in S$.

*Definition 5:* **(Right-isotonicity)** The quadruplet $(S,\oplus,w,\preceq)$ is right-isotonic if $w(a) \preceq w(b)$ implies $w(a \oplus c) \preceq w(b \oplus c)$, for all $a,b,c \in S$. Similarly, $(S,\oplus,w,\preceq)$ is strictly right-isotonic if $w(a) \prec w(b)$ implies $w(a \oplus c) \prec w(b \oplus c)$, for all $a,b,c \in S$.

*Definition 6:* **(Isotonicity)** The quadruplet $(S,\oplus,w,\preceq)$ is isotonic if $w(a) \preceq w(b)$ implies both $w(a \oplus c) \preceq w(b \oplus c)$

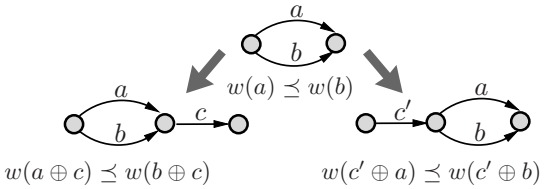


Fig. 3. Example of isotonicity

and $w(c' \oplus a) \preceq w(c' \oplus b)$, for all $a, b, c, c' \in S$. Similarly, $(S, \oplus, w, \preceq)$ is strictly isotonic if $w(a) \prec w(b)$ implies both $w(a \oplus c) \prec w(b \oplus c)$ and $w(c' \oplus a) \prec w(c' \oplus b)$, for all $a, b, c, c' \in S$.

Basically, left-isotonicity states that the order relation between the weights of any two paths is preserved if both of them are *prefixed* by a common, third path as shown in Figure 1. However, the order relation is NOT necessarily preserved if both of them are *appended* by a common, third path. Similarly, right-isotonicity states that the order relation between the weights of any two paths is preserved if both of them are *appended* by a common, third path as shown in Figure 2. But the order relation is NOT necessarily preserved if both of them are *prefixed* by a common, third path. It is easy to verify that isotonicity implies both left- and right-isotonicity, and the isotonicity holds if both left- and right-isotonicity hold. An example of isotonicity is shown in Figure 3.

*Definition 7:* **(Left-monotonicity)** The quadruplet $(S, \oplus, w, \preceq)$ is left-monotonic if $w(a) \preceq w(b \oplus a)$ holds for any paths $a$ and $b$.

*Definition 8:* **(Right-monotonicity)** The quadruplet $(S, \oplus, w, \preceq)$ is right-monotonic if $w(a) \preceq w(a \oplus b)$ holds for any path $a$ and $b$.

*Definition 9:* **(Monotonicity)** The quadruplet $(S, \oplus, w, \preceq)$ is monotonic if $w(a) \preceq w(a \oplus b)$ and $w(a) \preceq w(c \oplus a)$ holds for any paths $a$, $b$ and $c$.

Left-monotonicity implies that the weight of a path does not decrease when prefixed by another path. Right-monotonicity implies that the weight of a path does not decrease when suffixed by another path. Monotonicity implies both left- and right-monotonicity, and the monotonicity holds if both left- and right-monotonicity hold.

### *B. Isotonicity and Lightest Paths*

Isotonicity and monotonicity are important properties that determine the consistency, optimality and loop-freeness of different routing protocols. In Sobrinho's work [9], it has been proved that strict isotonicity and monotonicity are the sufficient and necessary condition for the existence of a *perfect lightest path* between any pair of nodes, where a perfect lightest path is a simple path such that all its subpaths are lightest paths on their own. With strict isotonicity and monotonicity, all the routing protocols discussed in this paper are guaranteed to be optimal, consistent and loop-free. However, for many routing protocols, perfect lightest paths are not always necessary to ensure optimal, consistent and loop-free routing. Hence, for these protocols, we can relax the requirements for path weight structures and enable more freedom in designing routing metrics to capture the characteristics of wireless networks. By dividing the isotonicity into the left and right properties, we have the following results regarding the relationship between isotonicity and lightest paths. These results are used in our analysis of different routing protocols in Sections IV, V and VI.

*Lemma 1:* 1. If the path weight structure $(S, \oplus, w, \preceq)$ is right-isotonic, for every $s, d \in V$, there exists a lightest path from $s$ to $d$ such that all its subpaths with source $s$ are also lightest paths. Such a lightest path is called a S-lightest path.

2. If the weight structure $(S, \oplus, w, \preceq)$ is strictly right-isotonic, all lightest paths are S-lightest paths.

*Proof:* The proof in part 1 of this lemma is similar to the one shown in [9]. The only difference is that we show that the theorem holds for right-isotonicity while the proof in [9] requires isotonicity. To prove part 1 of this lemma, we construct a sequence of paths $p^i = \langle v_1^i, v_2^i, \cdots, v_{n_i}^i \rangle$, $i \geq 1$, from $s = v_1^i$ to $d = v_n^i$, eventually leading to an S-lightest path from $s$ to $v$. Paths $p^i$ are constructed according to the following procedure. Initially, choose any path from $s$ to $v$ as path $p^1$. At step $i, i \geq 1$, if $p^i$ is an S-lightest path from $s$ to $v$ then stop. Otherwise, let $v_{k_i}^i, 1 < k_i \leq n_i$, be the last node along $p^i$ for which $\text{SUBPATH}(p^i, v_1^i, v_{k_i}^i)$ is not a lightest path. Choose $q^i$ to be any lightest path from $s$ to $v_{k_i}^i$ and let $p^{i+1} = q^i \oplus \text{SUBPATH}(p, v_{k_i}^i, v_{n_i}^i)$. Since $(S, \oplus, w, \preceq)$ is right-isotonic, $w(q^i) \prec \text{SUBPATH}(p^i, v_1^i, v_{k_i}^i)$ implies $w(p^{i+1}) \preceq w(p^i)$. In addition, $k_{i+1} < k_i$. Since the number of nodes in any path is bounded by $|V|$ and the $k_i$s decrease monotonically, the procedure stops after a finite number of steps, and when it does, we get an S-lightest path from $s$ to $v$.

We prove part 2 of this lemma by contradiction. Decompose the lightest path $p$ into $p = p' \oplus \text{SUBPATH}(p, v_k, d)$, where the subpaths of $p$ with origin at $s$ and destination at $v_i, i > k$ are lightest paths between $s$ and $v_i$, but the subpath $p' = \text{SUBPATH}(p, s, v_k)$ is not the lightest path between $s$ and $v_k$. There must exist another path $q = \langle s, u_1, u_2, \cdots, v_k \rangle$ such that $q$ is the lightest path between $s$ and $v_k$ and $w(q) \prec w(p')$. Since the weight structure is strictly right-isotonic, $w(q \oplus \text{SUBPATH}(p, v_k, d)) \prec w(p)$, which contradicts that $p$ is the lightest path between $s$ and $d$. ■

*Lemma 2:* 1. If the weight structure $(S, \oplus, w, \preceq)$ is left-isotonic, for every $s, d \in V$, there exists a lightest path from $s$ to $d$ such that all its subpaths with destination $d$ are also lightest paths. Such a lightest path is called a D-lightest path.

2. If the weight structure $(S, \oplus, w, \preceq)$ is strictly left-isotonic, all lightest paths are D-lightest paths.

*Proof:*

Similar to the proof of Lemma 1, to prove part 1 of this lemma, we construct a sequence of paths $p^i = \langle v_1^i, v_2^i, \cdots, v_{n_i}^i \rangle$, $i \geq 1$, from $s = v_1^i$ to $d = v_n^i$, eventually leading to a D-lightest path from $s$ to $v$. Paths $p^i$ are constructed according to the following procedure. Initially, choose any path from $s$ to $v$ as path $p^1$. At step $i, i \geq 1$, if $p^i$ is a D-lightest path from $s$ to $v$ then stop. Otherwise, let $v_{k_i}^i, 1 \leq k_i < n_i$, be the first node along $p^i$ for which $\text{SUBPATH}(p^i, v_{k_i}^i, d)$ is not a lightest path. Choose $q^i$ to be any lightest path from $v_{k_i}^i$ to $d$ and let $p^{i+1} = \text{SUBPATH}(p^i, s, v_{k_i}^i) \oplus q^i$. Since $(S, \oplus, w, \preceq)$ is left-isotonic, $w(q^i) \prec \text{SUBPATH}(p^i, v_{k_i}^i, d)$ implies $w(p^{i+1}) \preceq w(p^i)$. In addition, $k_{i+1} > k_i$. Since the number of nodes in any path is bounded by $|V|$ and the $k_i$s grow monotonically, the procedure stops after a finite number of steps, and when it does, we get an D-lightest path from $s$ to $v$.

We prove part 2 of this lemma by contradiction. Decompose the lightest path $p$ into $p = \text{SUBPATH}(p, s, v_k) \oplus p'$, where the subpaths of $p$ with destination at $d$ and origin at $v_i, i < k$ are lightest paths between $v_i$ and $d$, but the subpath $p' = \text{SUBPATH}(p, v_k, d)$ is not the lightest path between $v_k$ and $d$. There must exist another path $q = \langle v_k, u_1, u_2, \cdots, d\rangle$ such that $q$ is the lightest path between $v_k$ and $d$ and $w(q) \prec w(p')$. Since the weight structure is strictly left-isotonic, $w(\text{SUBPATH}(p, s, v_k) \oplus q) \prec w(p)$, which contradicts the assumption that $p$ is the lightest path between $s$ and $d$. ■

*Lemma 3:* If the weight structure $(S, \oplus, w, \preceq)$ is strictly isotonic, all subpaths of a lightest path are also lightest paths. In another word, all lightest paths are perfect lightest paths.

*Proof:* Assume that there is a lightest path $p = \langle s, v_1, v_2, \cdots, v_n, d\rangle$ between nodes $s$ and $d$. If a subpath $p' = \text{SUBPATH}(p, v_i, v_j)$ of $p$ is not the lightest path, there must exist another path $q = \langle v_i, u_1, \cdots, u_l, v_j, \rangle$ such that $w(q) \prec w(p')$. Since $(S, \oplus, w, \preceq)$ is strictly isotonic, $w(\text{SUBPATH}(p, s, v_i) \oplus q \oplus \text{SUBPATH}(p, v_j, d)) \prec w(p)$. Hence, we find a path between $s$ and $d$ that has a lighter weight than $p$, which contradicts with the assumption that $p$ is the lightest path between $s$ and $d$. ■

## IV. Weight Structure and Flooding-based Reactive Routing

To ensure the optimality of a routing scheme, efficient algorithms are needed to calculate lightest weight paths. In an ideal network that has no packet losses, the route calculation in reactive routing works as follows.

A source node that intends to communicate with a destination node initiates a network-wide flooding as follows. The source node first creates a route request message and puts its own address in the message. Then, the source node broadcasts this route request message to its neighboring nodes. Its neighboring nodes rebroadcast this route request message to their neighbors, eventually flooding the route request message to the entire network. During the flooding process, the route request message carries the necessary information for calculating the weight of the path that it has traversed. Depending on the goal of different routing protocols, such necessary information varies dramatically from one network to the other. Typical examples of such information include bottleneck link capacity, hop count, delay, remaining battery power and channel assignment along the path.

An intermediate relaying node may receive multiple route request messages coming from different paths. This intermediate node calculates the weight of paths that these route request messages have traversed and chooses to rebroadcast the route request message that traverses the lightest simple path. The route request messages that are not rebroadcast are dropped. If there are two or more route request messages that carry the same lightest weight, the intermediate node either rebroadcasts all of these route request messages or enforces a tie breaking scheme that assigns additional weights to the paths to ensure the uniqueness of the lightest weight path. The tie breaking scheme is essentially part of the weight function $w(\cdot)$ in the path weight structure.

When a copy of the route request message reaches the destination node, this copy traverses a simple path from the source to the destination. The destination node may receive multiple route request messages and it selects the one that traverses the lightest path and returns a route reply message to the source following this lightest path. The uniqueness of the reply path is enforced by using tie breaking schemes to assign additional weights to paths that originally have the same weight.

If the flooding-based route discovery is combined with hop-by-hop forwarding scheme (e.g. AODV [1]), the routing table entries at the nodes along the reply path are set when the route reply message travels back to the source node. If the flooding-based route discovery scheme is combined with source routing (e.g. DSR [2] and MCR [3]), the entire path information is carried back to the source node by the route reply message.

Given the above route discovery process in reactive routing protocols, we next study the properties of routing metrics to ensure the optimality, consistency and loop-freeness of reactive routing schemes. Table I summarizes the results of our analysis.

### A. Calculation of Lightest Paths

To satisfy the optimality requirement for any routing protocol, the path calculation algorithm of this protocol must be able to find the lightest path between any pair of nodes. In addition, the packet forwarding scheme of the routing protocol must consistently forward the packets along the lightest path. In this section, we focus on the requirements for calculating lightest path in flooding-based route discovery and Section IV-B discusses the optimality and consistency of reactive routing.

*Theorem 1:* Under the ideal case where broadcast packets are always reliably delivered, the flooding-based route discovery is guaranteed to find the lightest path between any pair of nodes if and only if the path weight structure is right-isotonic.

*Proof:*

*Sufficient condition:*

We prove the sufficient condition by contradiction. Suppose there are multiple paths between a source node $s \in V$ and a destination node $d \in V$. Assume the lightest path between $s$ and $d$ is $p = \langle s, v_1, v_2 \cdots, v_n, d\rangle$ and the route discovery process fails to find $p$. This can only happen when none of the route request messages that traverses the lightest paths reach the destination, which means that all of these route request messages are dropped at some intermediate relaying nodes. Suppose the last route request message that is relayed along $p$ is dropped at an intermediate node $v_k$. $v_k$ discards this route request message because $v_k$ has received another route request message coming from another path $q = \langle s, u_1, u_2 \cdots, u_m, v_k\rangle$ and $w(q) \prec w(\text{SUBPATH}(p, s, v_k))$. Since the weight structure is right-isotonic, we have $w(q \oplus \text{SUBPATH}(p, v_k, d)) \preceq w(p)$. If $w(q \oplus \text{SUBPATH}(p, v_k, d)) \prec w(p)$, it contradicts with the fact that $q$ is not a lightest path. If $w(q \oplus \text{SUBPATH}(p, v_k, d)) = w(p)$, $q \oplus \text{SUBPATH}(p, v_k, d)$ is a lightest path and $v_k$ is still

| **Routing protocols** | Optimality | Consistency | Loop-freeness |
|---|---|---|---|
| Flooding-based route discovery + source routing | right-isotonicity | | |
| Flooding-based route discovery + hop-by-hop routing | right-isotonicity + strictly left-isotonicity | right-isotonicity + strictly left-isotonicity | circle detection mechanism |
| Dijkstra's algorithm + source routing | right-isotonicity + right-monotonicity | | |
| Dijkstra's algorithm + hop-by-hop routing | right-isotonicity + right-monotonicity + strictly left-isotonicity | right-isotonicity + right-monotonicity + strictly left-isotonicity | right-isotonicity + right-monotonicity + strictly left-isotonicity |
| Distributed Bellman-Ford algorithm + source routing | left-isotonicity | | |
| Distributed Bellman-Ford algorithm + hop-by-hop routing | left-isotonicity + left-monotonicity | left-monotonicity | left-monotonicity |

TABLE I

RELATIONSHIP BETWEEN PATH WEIGHT STRUCTURE AND ROUTING PROTOCOLS

forwarding the route request message following a lightest path. This contradicts with the assumption that $v_k$ drops the last route request message that is forwarded along a lightest path.

*Necessary condition*

We prove the necessary condition through a simple example. In the topology in Figure 4, since the weight structure is not right-isotonic, it is possible that $w(a) \preceq w(b)$ but $w(b \oplus c) \prec w(a \oplus c)$. When two copies of a route request message from the source node A to the destination node C reach node B through two different paths $a$ and $b$, node B may only rebroadcasts the request message coming from path $a$ since $w(a) \preceq w(b)$. When node C receives the route request message from node B, node C decides that $a \oplus c$ is the lightest path, which is not correct since path $w(b \oplus c) \prec w(a \oplus c)$. Hence, right-isotonicity is necessary to ensure correct calculation of lightest path for flooding-based routing discovery. ■

### B. Consistency in Routing

In an ideal connected network where broadcast messages are always reliably delivered, a route discovery message initiated from a source node $s$ will eventually reach the destination node $d$ and a path between $s$ and $d$ can always be found by the flooding-based route discovery. Since source routing automatically guarantees consistent routing, based on Theorem 1, we can conclude that for routing protocols composed of flooding-based route discovery and source routing (e.g. DSR [2] and MCR [3]), both the optimality and consistency of routing are guaranteed if and only if the weight structure is right-isotonic.

However, if flooding-based route discovery is combined with hop-by-hop routing (e.g. AODV [1]), routing consistency imposes additional requirements on the properties of weight structure. This is because inconsistent routing may happen in the following situation. Assume that a source node $s$ selects a path $p = \langle s, v_1, v_2, \cdots, v_n, d \rangle$ to a destination $d$ through flooding-based route discovery. The routing table along the path $p$ has been set up appropriately to ensure that when $v_k$ receives a packet destined to node $d$, $v_k$ forwards the packet to $v_{k+1}$ until the packet reaches $d$. Assume that another source node $t$ wants to reach destination $d$ as well and starts another flooding-based route discovery. Denote the path selected by $t$'s route discovery as $q = \langle t, u_1, u_2, \cdots u_m, d \rangle$. If the path weight structure is not properly set, it is possible that paths $p$ and $q$ intersect at $v_i = u_j$ for some $i, j$ and $\text{SUBPATH}(p, v_i, d)$ is different from $\text{SUBPATH}(q, u_j, d)$. This means that when $t$ sets up the routing table along $q$, it rewrites the forwarding entry at node $v_i$. Therefore, the packet from node $s$ will be forwarded along path $q$ after it reaches $v_i$. Hence, the packet is not forwarded along the original lightest path $p$ that is selected by the source node $s$, which indicates inconsistent routing. To ensure consistent routing, we have the following theorem.

*Theorem 2:* If the routing table in hop-by-hop routing is setup using flooding-based route discovery, the routing is consistent if and only if the weight structure is both right-isotonic and strictly left-isotonic.

*Proof:*

*Sufficient condition:*

Assume that a source node $s$ initiates a flooding-based route discovery and find path $p$ as its path to a destination node $d$. Another source node $t$ discovers another path $q$ to the destination node $d$ through flooding-based route discovery. Since the weight structure is right-isotonic, both $p$ and $q$ are the lightest paths according to Theorem 1.

If paths $p$ and $q$ have no common nodes, obviously, $s$ and $t$ do not affect each other's routing consistency. If paths $p$ and $q$ intersect at a common node $r \neq d$, to show the routing consistency, we only need to prove $\text{SUBPATH}(p, r, d) = \text{SUBPATH}(q, r, d)$ as follows. Since $p$ is the lightest path between $s$ and $d$, at any node $v_k$ on $\text{SUBPATH}(p, r, d)$, we must have $w(\text{SUBPATH}(p, s, v_k)) \preceq w(\text{SUBPATH}(p, s, r) \oplus \phi(r, v_k))$, where $\phi(r, v_k)$ is any path from $r$ to $v_k$ that is different from $\text{SUBPATH}(p, r, v_k)$. Otherwise, $\text{SUBPATH}(p, s, r) \oplus \phi(r, v_k) \oplus \text{SUBPATH}(p, v_k, d)$ would be lighter than $p$, which contradicts the fact

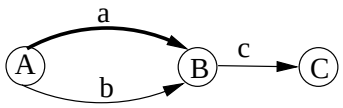


Fig. 4. Example topology for weight structure that lacks right-isotonicity. $w(a) \preceq w(b)$ but $w(b \oplus c) \prec w(a \oplus c)$

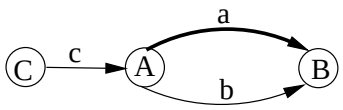


Fig. 5. Example topology for weight structure that lacks left-isotonicity. $w(a) \preceq w(b)$ but $w(c \oplus b) \prec w(c \oplus a)$

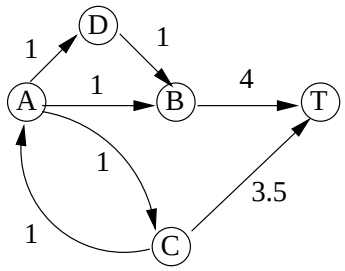


Fig. 6. Inconsistent routing and potential loops caused by lack of right-isotonicity

that $p$ is the lightest path between $s$ and $d$. Since the path weight structure is strictly left-isotonic, $w(\text{SUBPATH}(p, s, v_k)) \preceq w(\text{SUBPATH}(p, s, r) \oplus \phi(r, v_k))$ indicates that $w(\text{SUBPATH}(p, r, v_k)) \preceq w(\phi(r, v_k))$. Therefore, $w(\text{SUBPATH}(q, t, r) \oplus \text{SUBPATH}(p, r, v_k)) \preceq w(\text{SUBPATH}(q, t, r) \oplus \phi(r, v_k))$. Since $q$ is the lightest path between $t$ and $d$, $\text{SUBPATH}(p, r, v_k)) = \text{SUBPATH}(q, r, v_k)$.

*Necessary condition:*

If the weight structure is not strictly left-isotonic, we can construct a topology that has inconsistent routing. In Figure 5, since the weight structure is not strictly left-isotonic, it is possible that $w(a) \preceq w(b)$ but $w(c \oplus b) \prec w(c \oplus a)$. Hence, node C's route discovery selects path $c \oplus b$ as the lightest weight path between C and B but node A's route discovery selects $a$ as the lightest path between A and B, creating inconsistent routing decisions.

We prove the necessity of right-isotonicity through an example topology depicted in Figure 6, where the number by the side of each link is the weight of the link. The weight of a path $p = \langle v_1, v_2, \cdots v_n \rangle$ is defined by a weighted summation of link weights as $w(p) = \sum_{i=1}^{n-1} w(l_i)/2^{(i-1)}$, where $l_i$ is the link between node $v_i$ and $v_{i+1}$ on path $p$. This definition of weight structure is strictly left-isotonic and right-monotonic, but it is not right-isotonic. Using this weight structure, we have $\langle A, D, B, T \rangle \prec \langle A, C, T \rangle \prec \langle A, B, T \rangle$, $\langle A, B \rangle \prec \langle A, D, B \rangle$, $\langle C, A, B \rangle \prec \langle C, A, D, B \rangle$ and $\langle C, A, B, T \rangle \prec \langle C, T \rangle$. Given the above orders between path weights, the route discovery initiated by node A chooses $\langle A, C, T \rangle$ as the path to T and the route discovery initiated by node C chooses $\langle C, A, B, T \rangle$ as its path to T. Hence, the routing decisions in nodes A and B are inconsistent. Hence, right-isotonicity is necessary for consistent routing. ■

Based on Theorem 2, given the flooding-based route discovery, if the path weight structure is right-isotonic and strictly left-isotonic, routing is guaranteed to be both consistent and optimal for both source routing and hop-by-hop routing. If the path weight structure is right-isotonic, consistent and optimal routing can only be achieved through source routing.

### C. Loop-freeness in Routing

Ensuring loop-freeness in flooding-based reactive routing is very easy. For source routing, loop-freeness is automatically ensured since the source can remove any loop in the path. For hop-by-hop reactive routing, we have the following theorem.

*Theorem 3:* If the path selected by a flooding-based route discovery does not have any circle, this route discovery cannot create routing loops.

*Proof:* Assume that the route discovery selects a path $p = \langle v_1, v_2, \cdots v_n d \rangle$ between nodes $v_1$ and $d$. Since $p$ has no circle, if $p$ has no intersection with any other node's path to $d$, no routing loop is created by establishing path $p$ in the network. If $p$ intersects with another existing loop-free path $q = \langle u_1, u_2, \cdots u_m, d \rangle$, $p$ rewrites the routing entries to node $d$ at those intersection nodes. Given a packet starting anywhere in path $q$, the packet is forwarded along path $q$ until it either reaches the destination $d$ or reaches an intersection nodes of $p$ and $q$, If the packet meets one of the intersection nodes, it will then be forwarded along path $p$ until it reaches $d$. In both cases, there is no routing loop since both $p$ and $q$ are loop-free. ■

Theorem 3 essentially says that as long as the path selected is loop-free every time that a route discovery process is initiated, the hop-by-hop reactive routing protocol is guaranteed to be loop-free even if it is not consistent and optimal.

It is very easy to ensure that flooding-based route discovery only selects loop-free paths. One method is to check the path carried in the packet header of a route request message (e.g. DSR). If there is a circle in the path, a relaying node simply drops the route request message, essentially giving paths with circles a higher weight than circle-free paths. For route discovery schemes that do not put the entire path in route request message's header (e.g., AODV), loop-freeness can be ensured by path weight structure that is right-monotonic. Under right-monotonicity, if a route request message traverses a circle $\langle v_1, v_2, \cdots, v_n = v_1 \rangle$ starting at node $v_1$, the weight of its path does not decrease. If the weight of the path increases after this route request message traverses the circle, $v_1$ simply drops this route request message since $v_1$ only forwards the route request message from the lightest weight paths. If the weight of the path remains the same after the circle, tie breaking schemes based on the delay of paths can ensure that $v_1$ drops this route request message.

In conclusion, for flooding-based reactive routing, if source routing is used, the optimality of routing requires right-isotonicity and the consistency and loop-freeness do not have

specific requirements on path weight structure. If hop-by-hop routing is used, the optimality of routing requires right-isotonicity, the consistency requires both right-isotonicity and strictly left-isotonicity, and loop-freeness does not have specific requirements on path weight structure.

## V. Proactive Routing using Dijkstra's algorithm

A large group of proactive routing schemes relies on Dijkstra's algorithm to calculate the paths from a source node $s$ to all the other nodes in the network. The pseudo code of Dijkstra's algorithm is shown in Algorithm 1, where $N[u]$ is the neighboring nodes of node $u$, $p[v]$ stores the path from source node $s$ to node $v$ and $l[v]$ stores the weight of path $p[v]$. [1]

**Algorithm 1** Extended Dijkstra's algorithm.

**Dijkstra**$(G, w, s)$

1: **for** each node $v \in V$ **do**
2: $l[v] \leftarrow \infty$
3: $p[v] \leftarrow NIL$
4: **end for**
5: $Q \leftarrow V$
6: $l[s] \leftarrow 0$
7: $p[s] \leftarrow s$.
8: **while** $Q \neq 0$ **do**
9: $u \leftarrow Extract - Least - Elem(Q)$
10: **for** each node $v \in N(u)$ **do**
11: **if** $l[v] \succ w(p[u] \oplus (u, v))$ **then**
12: $l[v] \leftarrow w(p[u] \oplus (u, v))$
13: $p[v] \leftarrow p[u] \oplus (u, v)$
14: **end if**
15: **end for**
16: **end while**

In this section, we discuss what properties of path weight structure are required for ensuring the optimality, consistency and loop-freeness of routing protocols based on Dijkstra's algorithm.

### A. Calculation of Lightest Paths

Similar as the discussions in Section IV-A, the optimality of a routing protocol depends on whether the path calculation algorithm can find the lightest paths and whether the packet forwarding scheme is consistent. In this section, we focus on the requirements for path weight structure to ensure that Dijkstra's algorithm can find lightest paths. In Section V-B, we will discuss the properties of path weight structure to ensure routing consistency. Our discussion is similar to the work of Sobrinho [9] except that Sobrinho's work only shows that isotonicity and monotonicity are sufficient for calculation of lightest paths. Our work demonstrates that isotonicity requirement is too strict. In fact, just right-isotonicity and right-monotonicity are enough for calculation of lightest paths.

[1]The original Dijkstra's algorithm only uses hop-count as path weight. The algorithm in Algorithm 1 extends the original Dijkstra's algorithm so that it can work with any path weight structure.

*Theorem 4:* The necessary and sufficient condition for Dijkstra's algorithm to find the lightest paths is that the path weight structure must be right-isotonic and right-monotonic. The lightest paths found by Dijkstra's algorithm are also the S-lightest paths.

*Proof:*

*Sufficient condition:*

The proof is based on Algorithm 1. We first prove that, for each $v \in V$, we have $l[v] = \delta(s, v)$ at the time when $v$ is extracted from $Q$, where $\delta(s, v)$ is the weight of the lightest path from $s$ to $v$. Suppose otherwise, and let $v$ be the first node for which $l[v] \neq \delta(s, v)$ when it is extracted from $Q$. Obviously, $\delta(s, v) \prec l[v]$. According to Lemma 1, a S-lightest path, denoted as $p$, exists between $s$ and $v$. Decompose path $p$ into $p = p' \oplus \langle x, y\rangle \oplus p''$, where $y$ is the first node along $p$ that is still in $Q$ just before node $v$ is extracted, and $x$ is the predecessor of $y$.

Since $v$ is the first node to be extracted with $l[v] \neq \delta(s, v)$, when $x$ was extracted from $Q$, $l[x] = \delta(s, x)$. Since $p$ is the S-lightest path, $l[x] = \delta(s, x) = w(\text{SUBPATH}(p, s, x))$. In addition, we have $l[y] = w(\text{SUBPATH}(p, s, y)) = \delta(s, y)$ after the extraction of $x$ and subsequent relaxation of link $(x, y)$. In addition, due to right-monotonicity, $l[y] = w(\text{SUBPATH}(p, s, y)) \preceq w(p) = \delta(s, v) \prec l[v]$. This is a contradiction since when $v$ is extracted from $Q$, it is the node with the least lightest-path estimate of all nodes in $Q$. Hence, $l[v] = \delta(s, v)$ at the time when $v$ is extracted from $Q$.

From the code in Algorithm 1, it is easy to see that the lightest-path estimation $l[v]$ can only decrease during the execution of the algorithm. Once $l[v]$ reaches its lower bound $\delta(s, v)$, $l[v]$ and $p[v]$ remain unchanged. Hence, at the end of the execution of Dijkstra's algorithm, for any node $v$, $l[v] = \delta(s, v)$ and $p[v]$ holds the lightest path from $s$ to $v$.

To demonstrate that $p[v]$ is also the S-lightest path, we can make the following induction hypothesis. Suppose that during the execution of the Dijkstra's algorithm, for any node $t$ that is no longer in $Q$, $p[t]$ holds the S-lightest path between $s$ and $t$. For the base case, we have $l[s] = 0$ and $p[s] = NIL$ just after source $s$ is extracted from $Q$. For the induction step, consider a new node $v \neq s$ is extracted from $Q$. Let $u$ be $v$'s predecessor on $p[v]$. The induction hypothesis tells us that $p[u]$ is a S-lightest path. Since $l[v] = \delta(s, v) = w(p[u] \oplus (u, v))$, the path $p[v]$ is also the S-lightest path and remains unchanged afterwards.

*Necessary condition:*

We prove the necessity of right-isotonicity through a simple example. In the topology in Figure 4, since the weight structure is not right-isotonic, it is possible that $w(a) \preceq w(b)$ but $w(b \oplus c) \prec w(a \oplus c)$. Hence, when node B is extracted from the network, $l[B] = w(a)$ and the real lightest path $b \oplus c$ is not considered in Dijkstra's algorithm. Hence, $l[C] = w(a \oplus c)$ and the wrong path is selected as the lightest paths between nodes A and C.

We prove the necessity of right-monotonicity through another simple example. Assume that the weight structure is not right-monotonic but is right-isotonic. We construct a network

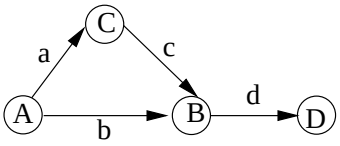


Fig. 7. Example topology for weight structure that is not right monotonic. $w(a) \prec w(b)$ and $w(b) \prec w(a \oplus c)$

as shown in Figure 7, where $w(a \oplus c) \prec w(b) \prec w(a)$. Since $w(b) \prec w(a)$, node B is extracted from $Q$ before node C. When B is extracted from Q, $l[D]$ is updated to $l[D] = w(b \oplus d)$ and this value of $l[D]$ remains unchanged. It is easy to see that $w(a \oplus c \oplus d) \prec l[D]$ due to right-isotonicity. Hence, Dijkstra's algorithm does not find the lightest path. ■

### B. Consistency

If source routing is used with Dijkstra's algorithm (e.g., LQSR [4] and HSLS [7]), obviously, routing is always consistent. This is because a source node can always use Dijkstra's algorithm to find a path to a destination even without isotonicity and monotonicity and no intermediate relaying nodes can change the route. Hence, based on Theorem 4, we can conclude that for source routing protocols based on Dijkstra's algorithm, the optimality of routing are guaranteed if and only if the path weight structure is right-isotonic and right-monotonic.

For hop-by-hop routing, consistent routing demands additional properties of path weight structure as shown in Theorem 5.

*Theorem 5:* Hop-by-hop routing based on Dijkstra's algorithm is consistent if and only if the path weight structure has the following three properties: right-isotonicity, strictly left-isotonicity and right-monotonicity.

*Proof: Sufficient condition:*

Right-isotonicity and right monotonicity ensures that Dijkstra's algorithm can always find the S-lightest path $p = \langle s, v_1, v_2, \cdots, v_n, d \rangle$ between a source node $s$ and a destination node $t$. Strictly left-isotonicity ensures that all the subpaths of $p$ with destination $d$ are all lightest paths. Hence, routing consistency is maintained.

Necessary condition:

The necessity of strictly left-isotonicity can be demonstrated by the example topology in Figure 5. In this figure, without strictly left-isotonicity, we can have $w(a) \prec w(b)$ and $w(c \oplus b) \preceq w(c \oplus a)$. In such case, Dijkstra's algorithm in node C can select path $c \oplus b$ as its lightest path to B, while Dijkstra's algorithm in node A selects $b$ as its lightest path to B, creating inconsistent routing.

Similar as the proof of Theorem 2, the necessity of right-isotonicity can be proved through the example topology depicted in Figure 6, where the weight of a path $p = \langle v_1, v_2, \cdots v_n \rangle$ is defined by a weighted summation of link weights as $w(p) = \sum_{i=1}^{n-1} w(l_i)/2^{(i-1)}$, where $l_i$ is the link between node $v_i$ and $v_{i+1}$ on path $p$. This definition of weight structure is strictly left-isotonic and right-monotonic, but it is not right-isotonic. Dijkstra's algorithm running in A chooses $\langle A, C, T \rangle$ as the path to T and Dijkstra's algorithm running in node C chooses $\langle C, A, B, T \rangle$ as its path to T. Hence, an inconsistent routing decision appears in nodes A and B.

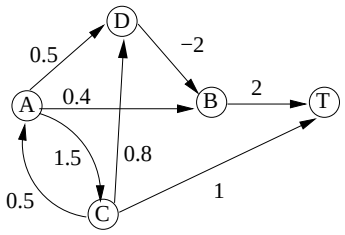


Fig. 8. Inconsistent routing and potential loops caused by lack of right-monotonicity

The necessity of right-monotonicity can be demonstrated through an example depicted in Figure 8. The weight of each link is represented by the numbers by the side of the link. The weight of a path is the summation of the weight of all its links. Since link $\langle D, B \rangle$ has a negative weight, the weight structure is not right-monotonic. It is easy to verify that Dijkstra's algorithm running in node A chooses path $\langle A, C, T \rangle$ to node T, while Dijkstra's algorithm running in node C chooses path $\langle C, A, D, B, T \rangle$ to node T. Hence, the routing choices at nodes A and C are inconsistent. ■

### C. Loop-freeness

For source routing using Dijkstra's algorithm, loop-freeness are ensured automatically without any particular requirement on path weight structure. For hop-by-hop routing based on Dijskstra's algorithm, to ensure loop-free routing, the path weight structure must be right-isotonic, strictly left-isotonic and right-monotonic, essentially having the same properties for ensuring consistent routing. These requirements for path weight structure are formally stated in Theorem 6 as follows.

*Theorem 6:* To achieve loop-free routing in Dijkstra-algorithm-based hop-by-hop routing protocols, the necessary and sufficient condition is that the weight structure has the following three properties: right-isotonicity, strictly left-isotonicity and right-monotonicity.

*Proof:* Sufficient condition:

Right-isotonicity ensures that S-lightest paths can always be found by Dijkstra's algorithm. Based on Theorem 5, consistent routing is ensured. Therefore, no routing loop can be created in the network.

Necessary condition:

We prove the necessity of strictly left-isotonicity by an example. For example, in Figure 9, assume the weight of a path be the weight of the first link on the path. The weight structure is right-isotonic and monotonic but not strictly left-isotonic. If $w(a) \prec w(c)$ and $w(b) \prec w(d)$, then Node B's lightest path to node C is $b \oplus c$ and node A's lightest path to node C is $a \oplus d$. Hence, when node B needs to send a packet to node C, the packet is forwarded to node A. Node A then forwards the packet back to node B, creating a routing loop between nodes A and B.

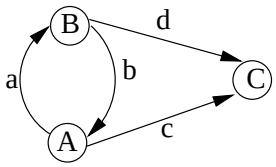


Fig. 9. Inconsistent routing in the lack of left isotonicity

The necessity of right-isotonicity and right-monotonicity can be proved using the proof in Theorem 5. In Theorem 5's proof for necessity of right-isotonicity, we have demonstrated that Node A chooses $\langle A, C, T\rangle$ as the path to T and node C chooses $\langle C, A, B, T\rangle$ as its path to T in Figure 6 due to the lack of right-isotonicity. Hence, a routing loop exists between nodes A and C.

Similarly, in Theorem 5's proof for necessity of right-monotonicity, we have demonstrated that Node A chooses $\langle A, C, T\rangle$ as the path to T and node C chooses $\langle C, A, B, T\rangle$ as its path to T in Figure 8. Hence, a routing loop exists between nodes A and C due to the lack of right-monotonicity. ■

## VI. Proactive Routing using Distributed Bellman-Ford Algorithm

Proactive routing protocols based on distributed Bellman-Ford algorithm are also named as distance vector protocols or path vector protocols. In this type of protocols, every node performs the following 3 step operations.

1) Each node calculates the current estimation of lightest paths between itself and all other nodes and stores this information as a table.
2) Each node sends its table to all neighbouring nodes.
3) When a node receives lightest path tables from its neighbours, it calculates the lightest paths to all other nodes and updates its own table to reflect any changes.

For routing protocols using the distributed Bellman-Ford algorithm, Sobrinho [10] has presented a very thorough work and the results are listed as Theorems 7, 8 and 9 in this section. The proofs of these theorems can be found in [10]. [2]

*Theorem 7:* The distributed Bellman-Ford algorithm can find the lightest path if and only if the path weight structure is left-isotonic.

Since source routing automatically guarantees consistency and loop-freeness, we can conclude that for source routing based on Bellman-Ford algorithm, the condition for optimality is left-isotonic. There is no specific requirements on path weight structure to ensure consistency and loop-freeness. For hop-by-hop routing based on the Bellman-Ford algorithm, the following theorems holds.

*Theorem 8:* To achieve consistent routing for hop-by-hop routing based on the distributed Bellman-Ford algorithm, the necessary and sufficient condition is that the path weight structure is left-monotonic.

[2]In [10], the terms "isotonicity" and "monotonicity" actually refer to the "left-isotonicity" and "left-monotonicity" concepts in this paper.

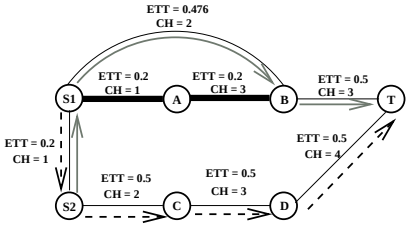


Fig. 10. Non-isotonicity of WCETT

*Theorem 9:* To achieve loop-free routing for hop-by-hop routing based on the distributed Bellman-Ford algorithm, the necessary and sufficient condition is that the path weight structure is left-monotonic.

## VII. Case Study

In the previous sections, we have analyzed the necessary and sufficient conditions for ensuring optimality, consistency and loop-freeness of different routing protocols. In this section, we utilize several existing routing metrics to demonstrate how to use our analytical results to design routing protocols.

### A. Case 1: WCETT

The first example is a path weight structure called WCETT, which was proposed by Draves et al. in [4] to reduce the number of nodes on the path of a flow that transmit in the same channel. The goal for WCETT is to reduce intra-flow interference. In WCETT, the weight of a path $p$ is defined as:

$$w(p) = (1-\beta)\sum_{\text{link } l\in p} ETT_l + \beta \max_{1\le j\le k} X_j, \qquad (1)$$

where $ETT_l$ is the expected transmission time of a packet over a link $l$, $\beta$ is a tunable parameter subject to $0 \le \beta \le 1$. $X_j$ is the number of times channel $j$ is used along path $p$ and captures the intra-flow interference. A path $p$ is lighter than a path $q$ ($w(p) \prec w(q)$) if $w(p)$ is smaller than $w(q)$.

WCETT is monotonic but is neither left-isotonic nor right-isotonic. Hence, we can conclude that none of the routing protocols that we have discussed so far can achieve optimal routing under WCETT. In addition, WCETT can achieve consistent and loop-free routing when source routing is used no matter what path calculation algorithm is used. For hop-by-hop routing, WCETT is loop-free only when the flooding-based route discovery or the distributed Bellman-Ford algorithm is used to calculate paths. For hop-by-hop routing, to ensure consistent routing under WCETT, the path calculation algorithm is further narrowed down to the distributed Bellman-Ford algorithm.

Figure 10 depicts a simple topology that demonstrate our conclusions. In this figure, two numbers are associated with each link, the ETT and the channel number (CH), respectively. Assuming $\beta$ in the definition of WCETT (see Equation (1)) is set to 0.5, the lightest path from $S_1$ to $T$ should be $S_1 \rightarrow B \rightarrow T$. However, due to the non-isotonic property of WCETT, when node $S_1$ uses Dijkstra's algorithm to calculate its path to node T, node $S_1$ incorrectly chooses $S_1 \rightarrow S_2 \rightarrow C \rightarrow D \rightarrow$

$T$ as the lightest path, indicated as the dotted arrows in Figure 10. This is because when running Dijkstra's algorithm at node $S_1$, the lightest path from node $S_1$ to node $B$ is found to be $S_1 \rightarrow A \rightarrow B$. $S_1 \rightarrow B$, hence, is eliminated from Dijkstra's algorithm's future consideration, although $S_1 \rightarrow A \rightarrow B \rightarrow T$ has a larger weight than $S_1 \rightarrow B \rightarrow T$. This incorrect early discard of $S_1 \rightarrow B$ causes Dijkstra's algorithm to fail to find the lightest path $S_1 \rightarrow B \rightarrow T$ from node $S_1$ to $T$.

This incorrect lightest path between $S_1$ and $T$ can cause forwarding loops if hop-by-hop routing is used. When node $S_2$ calculates its path to $T$, Dijkstra's algorithm correctly indicates that $S_2 \rightarrow S_1 \rightarrow B \rightarrow T$ is the lightest path, depicted as the shadowed arrows in Figure 10. Since $S_1$ has the incorrect lightest path, any packets destined to $T$ are forwarded by $S_1$ to $S_2$. $S_2$ immediately forwards the packets back to $S_1$ again. Hence, a forwarding loop is formed between $S_1$ and $S_2$.

Similar to Dijkstra's algorithm, routing protocols based on the Bellman-Ford algorithm (e.g., distance-vector routing) may not find lightest paths based on WCETT either. Using the same example in Figure 10, since node $B$'s lightest path to node $S_1$ is $B \rightarrow A \rightarrow S_1$, node $B$ only tells its neighbors about this path. Hence, node $T$ does not have a chance to check the weight of $T \rightarrow B \rightarrow S_1$, which is the correct lightest path. Therefore, node $T$ incorrectly sets its distance to $S_1$ as the weight of $T \rightarrow D \rightarrow C \rightarrow S_2 \rightarrow S_1$ and forwards any packets for $S_1$ to node $D$.

### B. Case 2: BAMER

BAMER [11] is a path weight structure proposed to capture the energy consumption for reliable communications over lossy wireless links. Under the BAMER path weight structure, the weight of a path $p$ between nodes $s$ and $v$ is recursively defined as:

$$w(p_{s,v}) = N(u,v) \times [w(p_{s,u}) + E(l_{u,v})], \qquad (2)$$

where $E(l_{u,v})$ is the energy consumption for a transmission over link $(u,v)$ and $N(u,v)$ is the expected number of transmissions of a successful delivery over link $(u,v)$ and $N(u,v) \geq 1$. The path weight $w(p_{s,v})$ essentially represents the energy consumption for successful delivering of a packet over path $p_{s,v}$ through end-to-end retransmissions. A path $p$ is lighter than a path $q$ ($w(p) \prec w(q)$) if $w(p)$ is smaller than $w(q)$.

The BAMER path weight structure is right-isotonic and monotonic but it is not left-isotonic. For example, in Figure 5, assume that $E(a) = 0.5, N(a) = 3, E(b) = 1, N(b) = 2, E(c) = 1, N(c) = 2$. It is easy to verify that $w(a) \prec w(b)$ but $w(c \oplus b) \prec w(c \oplus a)$. Hence, BAMER is not left-isotonic. Using our analytical results, we can easily see that BAMER is a perfect fit for source routing protocols combined with the flooding-based route discovery (e.g., DSR) or Dijkstra's algorithm. If BAMER is used with routing protocols based on the Bellman-Ford algorithm (e.g., DSDV), the routing protocol is not optimal, but the routing protocol is still consistent and loop-free. Therefore, BEAMER is still usable, although its performance may be degraded. If BEAMER is used with hop-by-hop routing based on Dijkstra's algorithm (e.g., link state protocols), the routing protocol cannot achieve optimality, consistency and loop-freeness. Hence, BAMER should never be used with hop-by-hop routing protocols based on Dijkstra's algorithm.

## VIII. Conclusions and Future Directions

In this paper, we have presented a comprehensive study of the design of path weight structure for wireless network routing protocols. We identify the three requirements, optimality, consistency and loop-freeness, for the proper operation of a variety of wireless routing protocols. We also analyze the relationship between isotonicity and monotonicity properties of path weight structure and the proper operation of routing protocols. The analytical results are summarized in Table I. Our work provides an important guideline for understanding the limitations of different routing metric design.

In the future, we would like to extend our analysis to more types of routing protocols, such as geographic routing, zone-based routing, etc. We will also focus on designing new routing protocols that has less strict requirements on path weight structure than existing protocols.

## References


[1] Charles Perkins, "Ad-hoc on-demand distance vector routing," in *MILCOM panel on Ad Hoc Networks*, 1997.

[2] David B Johnson and David A Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," in *Mobile Computing*. 1996, vol. 353, Kluwer Academic Publishers.

[3] Pradeep Kyasanur and Nitin Vaidya, "Multi-Channel Wireless Networks: Capacity and Protocols," Tech. Rep., University of Illinois at Urbana-Champaign, 2005.

[4] Richard Draves, Jitendra Padhye, and Brian Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *ACM Mobicom*, 2004.

[5] Guangyu Pei, Mario Gerla, Xiaoyan Hong, and Ching-Chuan Chiang, "Wireless Hierarchical Routing Protocol with Group Mobility," in *IEEE WCNC*, 1999.

[6] Thomas H. Clausen and Philippe Jacquet, "Optimized Link State Routing Protocol (OLSR)," IETF RFC 3626, October 2003.

[7] Cesar Santivanez and Ram Ramanathan, "Hazy Sighted Link State (HSLS) Routing: A Scalable Link State Algorithm," Tech. Rep. BBN Technical Memoradum No. 1301, BBN Technologies, 2001.

[8] Charles Perkins and Pravin Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *ACM SIGCOMM*, 1994.

[9] J. L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Inernet," in *IEEE INFOCOM*, 2001.

[10] J. L. Sobrinho, "Network Routing with Path Vector Protocols: Theory and Applications," in *ACM SIGCOMM*, 2003, pp. 49–60.

[11] Qunfeng Dong, Suman Banerjee, Micah Adler, and Archan Misra, "Minimum Energy Reliable Paths Using Unreliable Wireless Links," in *ACM Mobihoc*, 2005.