TIGHT: A Geographic Routing Protocol for Cognitive Radio Mobile Ad Hoc Networks

Xiaocong Jin, *Student Member, IEEE*, Rui Zhang, *Member, IEEE*, Jingchao Sun, *Student Member, IEEE*, and Yanchao Zhang, *Senior Member, IEEE*

***Abstract*—This paper presents TIGHT, a geographic routing protocol for cognitive radio mobile ad hoc networks. TIGHT offers three routing modes and allows secondary users to fully explore the transmission opportunities over a primary channel without affecting primary users (PUs). The greedy mode routes a packet via greedy geographic forwarding until a PU region is encountered and then further routes the packet around the PU region to where greedy forwarding can resume. It works best when the PUs are only occasionally active. In contrast, the optimal and suboptimal modes route a packet along optimal and suboptimal trajectories to the destination, respectively. They work best when the PUs are active most of the time. The suboptimal mode is computationally more efficient than the optimal mode at the cost of using suboptimal trajectories in rare cases. The efficacy of TIGHT is confirmed by extensive simulations.**

***Index Terms*—Cognitive radio, geographic routing, CR-MANET.**

I. Introduction

A COGNITIVE radio mobile ad hoc network (CR-MANET) refers to a MANET formed by mobile nodes with cognitive radios, which can be further divided into mobile primary users (PUs) and mobile secondary users (SUs) [1], [2]. There are a primary channel and also a secondary channel inferior to the primary one in terms of bandwidth, channel quality, etc. The PUs communicate over the primary channel, while the SUs mainly use the secondary channel and can opportunistically switch to the primary channel only in the regions free of PU communications. An exemplary CR-MANET of this kind is one deployed for military actions or post-disaster rescue, where the PUs and SUs correspond to VIP nodes coordinating the network mission (e.g., commanding vehicles or personnel) and average mobile nodes, respectively. Enabling cognitive communications in the above scenario can obviously help improve the spectrum efficiency of the primary channel and also the communication performance of the SUs.

Manuscript received January 18, 2014; revised April 2, 2014; accepted April 11, 2014. Date of publication April 28, 2014; date of current version August 8, 2014. This work was supported in part by the US National Science Foundation under grants CNS-0844972 (CAREER), CNS-1117462, and CNS-1320906. The associate editor coordinating the review of this paper and approving it for publication was Q. Li.

X. Jin, J. Sun, and Y. Zhang are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: xiaocong.jin@asu.edu; jcsun@asu.edu; yczhang@asu.edu).

R. Zhang is with the Department of Electrical Engineering, University of Hawaii, Honolulu, HI 96822 USA (e-mail: ruizhang@hawaii.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2014.2320950

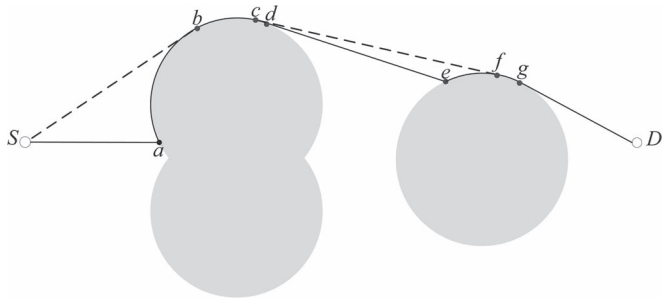

Fig. 1. Geographic routing in CR-MANETs, where only the source SU S and destination SU D are shown.

Effective multi-hop routing is a fundamental issue and challenge in CR-MANETs [3]. Existing routing protocols for conventional MANETs are not directly applicable to CR-MANETs, as there is the new requirement that the SUs make full use of the transmission opportunities over the primary channel which are induced by the PUs' mobility and dynamic communication activities. There have been some attempts (e.g., [4]–[9]) to adapt reactive MANET routing protocols such as AODV [15] and DSR [19] for CR-MANETs, which inevitably inherit some undesirable features of reactive MANET routing protocols. In particular, all involving route discovery and maintenance processes, these protocols require each node to maintain a lot of state information and thus are not scalable [10]. In addition, they are very sensitive to node mobility which may cause frequent route breakage and excessive routing control packets for repairing and rediscovering end-to-end routes [10].

Stateless geographic routing is more promising for CR-MANETs. Consider Fig. 1 as an example, where the shadowed regions represent where the SUs cannot use the primary channel and are called *PU region*s hereafter.[1] There are two intuitive strategies for geographically routing packets from S to D. First, if S knows D's location only, it can label every packet with D's location and then route it with classical GPSR [10] until the packet reaches the perimeter of the first PU region. Next, the packet is routed along the perimeter of the PU region and then returned to the GPSR mode. This process continues until the packet reaches D, and the ideally shortest physical path traversed by the packet comprises vector $\overrightarrow{Sa}$, arc $\overset{\frown}{ad}$, tangent vector $\overrightarrow{de}$, arc $\overset{\frown}{eg}$, and tangent vector $\overrightarrow{gD}$. The second strategy assumes that S additionally knows the PUs' locations whereby to construct the PU regions. Then S derives the shortest physical path towards D, consisting of tangent vector

[1] How a PU region is defined will be discussed shortly.

$\overrightarrow{Sb}$, arc $\overset{\frown}{bc}$, tangent vector $\overrightarrow{cf}$, arc $\overset{\frown}{fg}$, and tangent vector $\overrightarrow{gD}$. This latter physical path is obviously shorter than the first one. Both strategies only require each node to know its own location and a location service such as [12] for answering the queries about node locations. Since there is no routing table maintained at every node, they are stateless and scalable. In addition, they are relatively insensitive to node mobility and have much lower routing overhead for two reasons: (1) packet routing relies on node locations instead of particular next-hop nodes, and (2) there are no route discovery or maintenance.

This paper presents the design and evaluation of TIGHT, a novel geographic routing protocol for CR-MANETs to fully implement the above geographic routing strategies. The design and name of TIGHT comes from a simple observation. If there is no PU region blocking vector $\overrightarrow{SD}$, it is apparently the shortest trajectory over the primary channel from S to D. Now if we view the PU regions as physical obstacles, the shortest trajectory can be found by simply tightening a rope from S to D around the perimeters of some specific PU regions, and it will consist of a vector from S and tangent to the first PU region involved, a vector towards D and tangent to the last PU region involved, bitangent vectors with each involving two PU regions (if any), and the arcs connecting (bi)tangent line segments on the perimeters of involved PUs. TIGHT has three routing modes based on the above observation.

- *Greedy mode*: Corresponding to the first routing strategy above, this mode requires each packet forwarder (including the source itself) to perform local spectrum sensing to identify any nearby PU region and (if any) route around it based on our observation above: the shortest trajectory circumventing the PU region is along its perimeter until where greedy geographic forwarding can resume. This mode works best when the PUs are known to be only occasionally active and otherwise may use unnecessarily long trajectories (as shown in Fig. 1).
- *Optimal mode*: Corresponding to the second strategy above, this mode requires the source SU to compute the optimal (i.e., shortest) trajectory to the destination SU based on the PU locations. This mode works best when the PUs are known to be very active. The major challenge is on fast determination of the optimal trajectory for every packet, for which we propose an efficient solution.
- *Suboptimal mode*: This mode is proposed to further reduce the computational complexity of the optimal mode in computing per-packet trajectories at the expense of possibly deriving suboptimal trajectories.

The high performance of TIGHT is confirmed by extensive simulations. In particular, our simulation results show that, in contrast to classical GPSR involving the secondary channel only, TIGHT can dramatically reduce the end-to-end packet latency in simulated scenarios while maintaining comparable packet delivery ratio and thus confirm the great benefits of cognitive MANET communications.

The paper is organized as follows. Related work are summarized in Section II. Network model is discussed in Section III. We detail TIGHT algorithm design in Section IV. After that, we demonstrate the efficacy of the proposed scheme in Section V. In the final section, we draw the conclusion.

II. Related Work

This section discusses some work most germane to TIGHT, and we refer readers to [3] for a more comprehensive survey on routing in cognitive radio ad hoc networks.

Geographic routing protocols for conventional MANETs motivate the TIGHT design. GFG [17], [18] is a geographic routing algorithm that extracts a connected planar subgraph on which routing is performed. The packet is guaranteed of delivery as long as the unit graph is static and connected during the time to route a message. As the detailed routing protocol of GFG routing algorithm, GPSR [10] supports a greedy mode in which every node, say X, selects the next hop as the neighbor which is closest and also closer than X to the destination, say D. If a valid next hop exists, the greedy mode continues until the packet reaches D. If no such node exists, the perimeter mode of GPSR is activated to route the packet around the void until the greedy mode can resume. TBR [11] is another known geographic routing strategy in which every packet is routed along a physical trajectory to D. GPSR and TBR are nearly stateless, highly scalable, and very insensitive to node mobility. However, since GPSR does not take the PU information into account, directly adopting GPSR to CR-MANETs often leads to longer routing path. Also, how to obtain the optimal trajectory required by TBR in CR-MANETS remains an open challenge.

Most of the routing protocols proposed for CR-MANETs so far involve on-demand route discovery and thus require each intermediate node to maintain significant amount of state information. In particular, existing protocols [4]–[6], [8], [9] are adaptations of AODV [15] or DSR [19] to CR-MANETs. They all involve on-demand route discovery to find spectrum-aware end-to-end routes as well as route maintenance in case of route breaks due to node mobility or dynamic spectrum availability. SEARCH [7] sends on-demand route requests on every channel using greedy geographic routing, based on which the destination can then derive an optimal, possibly multi-channel path which involves multiple *anchor node*s with each identifying a unique PU region to be circumvented. As pointed out in [3], SEARCH's path optimization is very sensitive to node mobility and spectrum dynamics. In contrast, our TIGHT protocol is free of route discovery and maintenance, and it inherits the aforementioned merits of GPSR and TBR.

III. Network Model

We consider a CR-MANET with $N_{\rm p}$ PUs and $N_{\rm s}$ SUs with $N_{\rm p} \ll N_{\rm s}$. There are a primary channel of rate $R_{\rm p}$ and a secondary channel of rate $R_{\rm s} < R_{\rm p}$. Every PU communicates with the SUs and other PUs over the primary channel only. In contrast, every SU has a single transceiver tunable onto the secondary and primary channels, but it can use the primary channel only when the PUs are not interrupted. The extension of TIGHT to multiple primary/secondary channels is feasible and left as future work. Furthermore, we assume that the PUs do not participate in packet forwarding for simplicity only.

PU regions are defined as follows. Assume that the PUs and SUs have the same circular transmission range R. To ease the presentation, we also assume that every PU only communicates with the nodes (SUs or PUs) within its transmission range and discuss how to relax it in Section IV-D. It follows that all the SUs within $2R$ from any PU should not use the primary channel unless they are communicating with the PU. We thus define a *simple* PU region as a circle of radius $2R$ centered at a PU and a *complex* PU region as the union of multiple simple PU regions, each overlapping with at least one other simple PU region in the same complex PU region. Our subsequent illustrations involving a PU region without specifying the adjunct *simple* or *complex* apply to both a simple and a complex region. Furthermore, we assume that every SU can reliably detect existing PU transmissions by energy detection [20], feature detection [21], or other techniques [22]–[24].

It is worth noting that a PU may not always communicate. For example, it is commonly assumed that the communication activities of the PUs can be modeled as i.i.d. exponential ON/OFF processes with mean $\mu_{\rm ON}$ for the ON state and $\mu_{\rm OFF}$ for the OFF state. Therefore, a PU region defined based on the PU location(s) may be too conservative when $\mu_{\rm ON}$ is small, and/or $\mu_{\rm OFF}$ is large. How the PU activities affect our TIGHT protocol will be discussed shortly.

Similar to any geographic routing protocol such as GPSR [10] or TBR [11], TIGHT assumes that all the PUs and SUs know their own locations either from an internal GPS receiver or other positioning algorithms. In addition, both PUs and SUs can freely move, and their locations can be updated with and queried from an in-network location service such as [12]–[14]. The practicality of a scalable and lightweight location service has also been adequately argued in Section 3.7 in [10].

IV. TIGHT Design

This section details the TIGHT design. We first discuss the greedy, optimal, and suboptimal modes in sequel. Then we discuss some measures to enhance the performance of TIGHT. For convenience, we use S and D as the exemplary source and destination SUs henceforth. Since we assume that only the SUs participate in packet forwarding, the forwarding nodes involved in our following illustrations concern the SUs only.

A. Greedy Mode of TIGHT

Similar to GPSR [10], the greedy mode routes every packet based on the locations of every node's intermediate neighbors. Specifically, S marks every packet with D's location and then makes a locally optimal, greedy choice for the packet's next hop, which is the neighbor geographically closest to D. Every intermediate node first tries to make the same greedy choice, and the *void-handling* strategy of GPSR is invoked to route around any void encountered.

TIGHT's greedy mode differs from GPSR in the following aspects: First, the centroid of PU regions is defined to enable routing along shorter paths, also reducing the possibilities of routing failures; Second, the criterion of choosing the next hop node in our PU avoidance phase and the perimeter mode in GPSR is different. Compared to GPSR, TIGHT is more efficient and reliable in routing around PU regions using the proposed greedy mode.

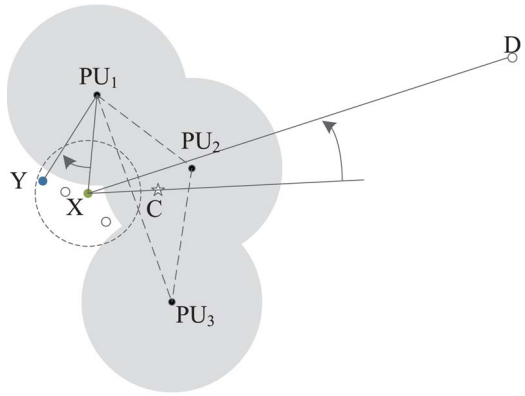

Fig. 2. Illustration of TIGHT's greedy mode, where C denotes the centroid of the complex PU region involving three PUs.

Below we explain with Fig. 2 to illustrate how to route around a PU region in TIGHT's greedy mode. The PU avoidance phase starts when a packet arrives at node X which is the closest SU to D outside the shown PU region. Even if X may have neighbors closer to D inside the PU region, the greedy forwarding of this packet has to stop because the closer SUs are not allowed to transmit over the primary channel. We address this issue by a simple geometric rule. Assume that X is aware of the locations of three nearby PUs, e.g., by exchanging information with neighboring nodes within the PU region over a common control channel. It then derives the centroid of the PU region, denoted by C, and checks whether the counterclockwise angle $\angle DXC$ is no larger than π. If so (as in Fig. 2), the physical path in the clockwise direction around the PU region is more likely to be the shorter one towards D. Then X appends to the packet its location and bit 0 which indicates clockwise. Assume that $\rm PU_1$ is the closest PU to X in the PU region, X forwards the packet to node Y, which is its most clockwise neighbor outside the PU region with respect to the vector $\overrightarrow{\rm PU_1X}$. Similarly, node Y determines its closest PU in the PU region (still $\rm PU_1$ in Fig. 2) and continues forwarding the packet to its most clockwise neighbor with respect to the vector from the corresponding PU to Y. During this PU avoidance process, the current forwarding node might move closer to another PU that is not marked for routing around previously, in this case, the node still moves according to the direction specified in the packet header but the current PU to route around is updated to be the closest PU. This process continues until the packet arrives at a node outside the PU region and closer to D than X. This node removes X's location and bit 0 from the packet, terminates the PU avoidance phase, and resumes the greedy forwarding. If the clockwise angle $\angle DXC$ is larger than π, X appends to the packet its location and bit 1 which indicates counterclockwise, and the packet is accordingly routed counterclockwise with respect to $\overrightarrow{\rm PU_1X}$. Also note that the packet may enter and exit the PU avoidance phase multiple times when circumventing the same PU region. It can be seen that the physical trajectory taken by the packet closely approximates the perimeter of the PU region if SU density is sufficiently high, which is in accordance with

our aforementioned observation for a tightened rope. Directly adopting the perimeter mode in GPSR is insufficient for the following reasons. First, when packets arrive at X, no clear instructions are given as to which direction to route around the PU regions. This could cause inefficiency, leading longer forwarding paths, and even forwarding failure; Second, The strategy of choosing the next hop node during the PU avoidance phase is still traversing progressively along closer faces towards D, hoping to find an exit in the end. In the case shown, it's possible that no closer faces can be found. Thus, it's essential to adopt the proposed greedy mode to tackle routing around PU regions.

B. Optimal Mode of TIGHT

As said, TIGHT's optimal mode can route every packet along the shortest (optimal) trajectory from S to D based on the known PU locations. It is motivated by TBR [11], and our contribution is an efficient way to determine the shortest trajectory in CR-MANETs. This mode consists of *trajectory computation* and *trajectory-based forwarding*.

1) Trajectory Computation: Assuming that the line segment $\overrightarrow{SD}$ is blocked by at least one PU region, the shortest trajectory then should circumvent a sequence of non-repeated PU regions. According to our tightened-rope observation, it intuitively consists of a tangent line segment from S to the first PU region involved, a tangent line segment from the last PU region involved to D, some line segments with each bitangent to two consecutive involved PU regions, and some arcs along the perimeters of involved PU regions. A line segment involved is tangent if involving S or D and bitangent if involving two PU regions. We thus shall not use the term bitangent henceforth to simplify the presentation. How could S efficiently derive the shortest trajectory to D based on the PU locations? For this purpose, we assume that the network region is enclosed in the upper-right quadrant of a Cartesian coordinate plane whose origin is the intersection of the lines tangent to the leftmost and bottommost points of the network region. The location of every node is then converted into the corresponding Cartesian coordinates. Since the shortest trajectory can be completely described by specific tangency points in order on some PU regions' perimeters, our goal is then to find these specific tangency points.

A plausible solution is to enumerate all possible trajectories and then pick the shortest. Assume that the vector $\overrightarrow{SD}$ is blocked by at least one PU region. Given $N_{\rm p}$ PUs, there are at most $N_{\rm p}$ non-overlapping PU regions, each being a simple region as a circle of radius $2R$ around a PU. The number of ways for the trajectory to involve $n \geq 1$ PU regions is thus up to $N_{\rm p}!/(N_{\rm p}-n)!$. There are two outer tangent line segments and two inner ones between any two consecutive simple PU regions, and there are also two tangent line segments from S to the first simple PU region and two tangent line segments from the last simple PU region to D. The number of candidate trajectories is thus up to $\#_{\rm total} = \sum_{n=1}^{N_{\rm p}} \#_n$, where $\#_n = 2 \times 2 \times 4^{n-1} \times (N_{\rm p}!/(N_{\rm p}-n)!) = (4^n N_{\rm p}!)/(N_{\rm p}-n)!$. This method is computationally expensive (if not infeasible) even for reasonably large $N_{\rm p}$. For example, $\#_{\rm total} \approx 4.88 \times 10^{12}$ for $N_{\rm p} = 10$, and $\#_{\rm total} \approx 3.43 \times 10^{30}$ for $N_{\rm p} = 20$. In addition, the shortest trajectory discovered with this method may be infeasible. For instance, a tangent line segment between two random PU regions may traverse some other PU regions, and any candidate trajectory containing such tangent line segments is infeasible.

Our solution depends on a directed weighted *forward graph* built upon on selected tangent line segments and the arcs connecting them. Specifically, let the vector $\overrightarrow{SD}$ define the forward direction. Then we convert every nondirectional tangent line segment, say $\overline{xy}$, into a vector $\overrightarrow{xy}$ if $\overrightarrow{xy}$ forms an acute angle with $\overrightarrow{SD}$ and into $\overrightarrow{yx}$ otherwise. Likewise, a nondirectional arc $\widehat{xy}$ is converted into a directional forward arc from x to y if $\overrightarrow{xy}$ forms an acute angle with $\overrightarrow{SD}$ and from y to x otherwise. Every vertex of the forward graph corresponds to a unique forward tangent line segment and has a weight equal to the segment length. An edge from one vertex to another corresponds to a forward arc connecting the two corresponding line segments, and the edge cost equals the arc length. S and D are also vertices of the forward graph, as they have outgoing and incoming tangent line segments, respectively. Once the forward graph is constructed, we can apply the classical Dijkstra algorithm to find the shortest path (trajectory) from S to D. Our solution is in line with the tightened-rope observation: the shortest trajectory from S to D consists of forward tangent line segments and arcs on a sequence of non-repeated PU regions.

The forward graph is constructed in three steps below.

Step 1) We eliminate the forward tangent line segments traversing at least one PU region, as forwarding packets along those segments will most likely interfere with the PUs. Since a simple PU region is a circle of radius $2R$ around a PU, those segments less than $2R$ from any PU are eliminated.

Step 2) We determine the edges of the forward graph, which is equivalent to deciding whether and how any two remaining forward tangent line segments can be connected. Consider two arbitrary forward line segments, say $\overrightarrow{ab}$ and $\overrightarrow{cd}$. A directional edge from $\overrightarrow{ab}$ to $\overrightarrow{cd}$ exists in the forward graph if and only if three following conditions are all satisfied.

1) $\overrightarrow{ab}$ and $\overrightarrow{cd}$ generally involve three distinct PUs in the forward direction, say PU_1, PU_2, and PU_3, with a on PU region-1, b and c on PU region-2, and d on PU region-3. The exception occurs when a corresponds to source S, and/or d corresponds to destination, in which case PU region-1, PU region-3, or both are undefined. This condition is necessary, as the shortest trajectory should not traverse the same two consecutive PU regions twice according to the tightened-rope observation.

2) $\overrightarrow{ab}$ and $\overrightarrow{cd}$ can be connected by a forward arc. A counter example is given in Fig. 3(a) for clarity. As we can see, if both $\overrightarrow{ab}$ and $\overrightarrow{cd}$ are on the shortest trajectory, we can only connect them via the arc from b to c which is not in the forward direction. This connection method, however, violates the tightened-rope observation.

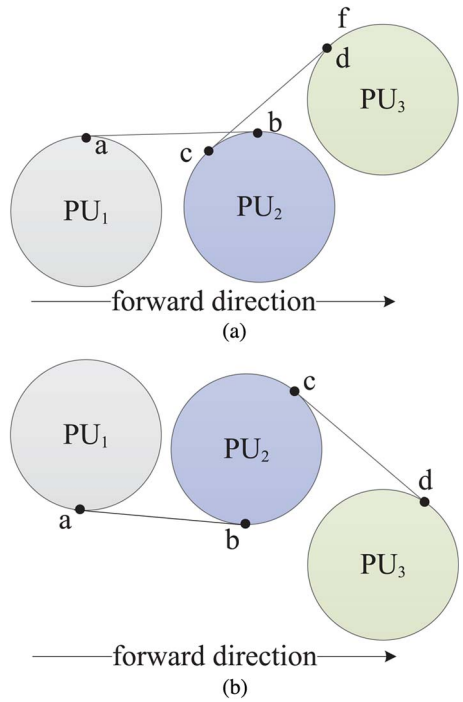

Fig. 3. Invalid combinations of tangent line segments.

3) Points b and c are both in the clockwise or counterclockwise directions of $\overrightarrow{\text{PU}_1\text{PU}_2}$ and $\overrightarrow{\text{PU}_2\text{PU}_3}$, respectively. We can easily see why this requirement is necessary through the counter example in Fig. 3(b), where b is clockwise from $\overrightarrow{\text{PU}_1\text{PU}_2}$, and c is counterclockwise from $\overrightarrow{\text{PU}_2\text{PU}_3}$. Since $\overrightarrow{ab}$ and $\overrightarrow{cd}$ can be connected by a forward arc from b to c, Condition 2 is thus satisfied. $\overrightarrow{ab}$ and $\overrightarrow{cd}$, however, cannot be both on the shortest trajectory (or a tightened rope from S to D around some PU regions).
4) The forward arc connecting $\overrightarrow{ab}$ to $\overrightarrow{cd}$ does not enter any PU region, which should be an obvious requirement.

If it is determined that a directional edge from $\overrightarrow{ab}$ to $\overrightarrow{cd}$ exists in the forward graph, the edge cost is simply the length of the forward arc connecting them.

Step 3) Recall that there are two forward tangent line segments from S to every simple PU region and two from every simple PU region towards D. To accommodate those segments surviving Step 1 and Step 2, we add to the forward graph two special vertices with zero weight for S and D. The vertex for S has outgoing edges of zero weight to its remaining line segments (i.e., vertices), and the vertex for D has incoming edges of zero weight from its remaining line segments (i.e., vertices). Finally, we remove the vertices (except the two special vertices) with no outgoing or incoming edge, and the forward graph is ready as input into the Dijkstra algorithm.

Computational Complexity Analysis

The computational complexity of the above operations is determined by the number of vertices in the forward graph. Specifically, the Dijkstra algorithm is known to run in $O(N^2)$ for a graph with N vertices. In our case, there are at most N_p PU regions with each being a simple PU region. So there are at most $2N_p$ forward tangent line segments from S and $2N_p$ to D, and there are two external and internal tangent line segments between every two PU regions. It follows that the forward graph contains no more than $N = 4N_p + 4\binom{N_p}{2}$ vertices. For example, N equals 220 for $N_p = 10$ and 840 for $N_p = 20$. In addition, $N \leq 4N_p + 2N_p^2$ for very large N_p, leading to the computational complexity of $O(N_p^4)$ for employing the Dijkstra algorithm. Building the graph itself has a higher requirement of computation capability due to the check of the existence of edges between pairs of vertices and also the requirement of arcs not entering any PU region. Thus, the computation complexity of building the graph is $O(N_p^5)$, which also represents the overall complexity for the algorithm. A method to reduce the computation complexity is deferred to Section IV-C.

An Example

A simple example is shown in Fig. 4(a) for clarity, where the tangent line segments blocked by any PU region (e.g., those from S to PU region-4) are not shown. Segment 2 is eliminated because the forward arcs connecting it to segments 9, 6, and 24 all enter PU region-1. Segment 22 is removed because it fails the third condition in Step 2: a_8 is in the counterclockwise direction of $\overrightarrow{S\text{PU}_1}$, while a_7 is in the clockwise direction of $\overrightarrow{\text{PU}_1\text{PU}_2}$. After segment 22 is removed, there is no other tangent line that can form a valid connection with segment 5, so segment 5 is also removed. Segments 9 and 24 are eliminated for lack of a valid connection to them from S. Segments 19, 10, and 15 are removed because there is no valid connection from them to D. The remaining segments can all be processed likewise, and more invalid tangent segments will be removed. Fig. 4(b) shows the eventual directed graph, where all the vertex and edge costs are determined during the graph construction.

2) Trajectory-Based Forwarding: Once the optimal trajectory comprising line and arc segments is decided, S inserts it in the packet header and starts trajectory-based forwarding. TIGHT adopts a greedy forwarding strategy: most advancement along the current segment in the trajectory. Specifically, S or every intermediate node tries to select the next hop as its neighbor closest to the end of the first segment and also outside of any PU region. If there are candidate next hops beyond the current segment, the one closest to the end of the next segment is chosen, in which case the current segment information is removed from the packet header, and the next segment becomes the current one. A void occurs if the current node cannot find a neighbor closer to the end of the current or next segment than itself. We can adopt the many void-handling strategies surveyed in [16] to route around the void.

C. Suboptimal Mode of TIGHT

This suboptimal mode is proposed to reduce the computational complexity of the optimal mode by building a different

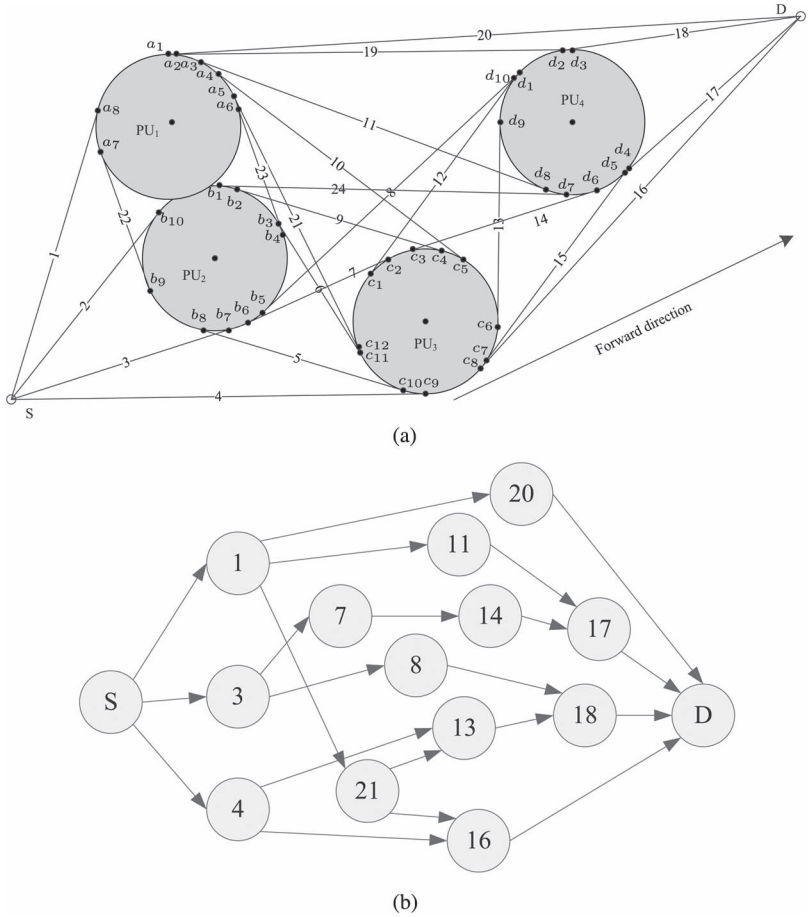

Fig. 4. An example for the forward graph.

forward graph with much fewer vertices. Recall that the shortest trajectory can be completely described by specific tangency points on a sequence of non-repeated PU regions. Consider any two PUs, say PU_1 and PU_2, where $\overrightarrow{\text{PU}_1\text{PU}_2}$ is in the forward direction., i.e., forming an acute angle with $\overrightarrow{SD}$. If PU region-1 does not overlap with PU region-2, there are two external tangency points and two internal ones on each PU region; otherwise, there are only two external ones on each PU range. We classify the tangency points on both regions in the counterclockwise and clockwise directions from $\overrightarrow{\text{PU}_1\text{PU}_2}$ into a LEFT group and a RIGHT group, respectively. Now consider PU_2 and PU_3, where $\overrightarrow{\text{PU}_2\text{PU}_3}$ is in the forward direction. The tangency points with regard to PU_2 and PU_3 can be similarly classified into a LEFT group and a RIGHT group. Since the forward direction $\overrightarrow{SD}$ is given, it can be easily shown that the LEFT (RIGHT) points on PU_2 with regard to $\overrightarrow{\text{PU}_1\text{PU}_2}$ are also LEFT (RIGHT) with regard to $\overrightarrow{\text{PU}_2\text{PU}_3}$. This observation can be generalized to any PU pairs in the forward direction. There are up to $4N_p$ tangency points on each PU region-x, including two for S, two for D, and four for each other PU. These $4N_p$ tangency points can be simply classified into two groups, denoted by LEFT_x and RIGHT_x, respectively.

The forward graph in the suboptimal mode is built upon the above groupings of tangency points. Specifically, every PU contributes two vertices to the forward graph, corresponding to LEFT_x and RIGHT_x. It is a little tricky to define the cost

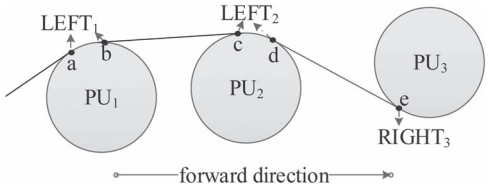

Fig. 5. Computing edge costs in the suboptimal mode, where the arrow indicates the group belonging of a tangency point.

of every directional edge. Consider Fig. 5 as an example, where not all the tangency points are shown for clarity. Assume that the shortest path from S to the vertex LEFT_1 has been discovered via the Dijkstra algorithm and reaches PU region-1 through a tangent line segment ending at point a which belongs to LEFT_1. Point a then determines a unique shortest trajectory from LEFT_1 to LEFT_2, comprising the arc from a to b and tangent vector $\overrightarrow{bc}$. Consequently, the cost of the edge from LEFT_1 to LEFT_2 is updated from undefined to $|\overset{\frown}{ab}| + |\overrightarrow{bc}|$. Likewise, if the shortest path from S to LEFT_2 can be discovered through $\overrightarrow{bc}$, the cost of the edge from LEFT_2 to RIGHT_3 is updated from undefined to $|\overset{\frown}{cd}| + |\overrightarrow{de}|$. In summary, an edge cost in the forward graph stays undefined until the shortest path from S to the starting vertex is discovered. Also different from the optimal mode, there is no cost associated with every vertex in the suboptimal mode. In addition, two virtual vertices are added to

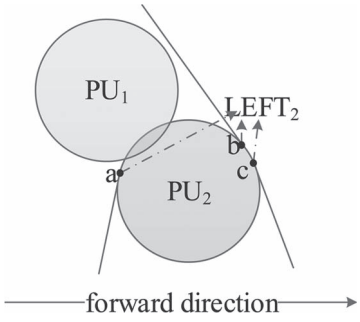

Fig. 6. Illustration of the modified Dijkstra algorithm.

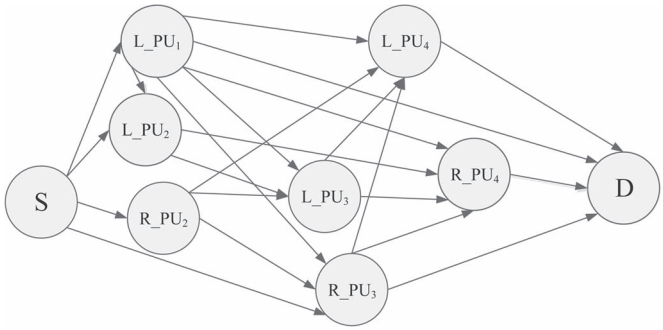

Fig. 7. Forward graph for Fig. 4(a) in suboptimal mode.

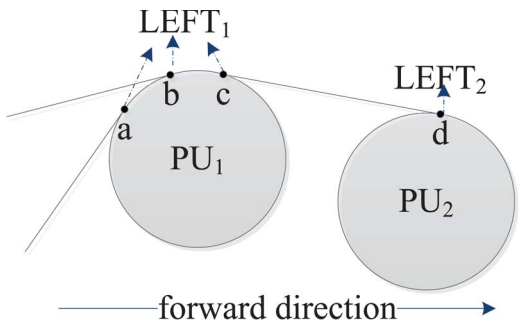

Fig. 8. Suboptimal trajectories in the suboptimal mode.

the forward graph for S and D, respectively, and every related edge cost is simply the length of the corresponding tangent line segment. Similar as in the optimal node, we need to determine whether any tangent line segment or any combination of an arc and a tangent line segment is valid. Specifically, every tangent line segment blocked by any PU is removed before algorithm execution. In addition, no arc in the backward direction or with two end points in the LEFT and RIGHT groups, respectively, can be used to define an edge cost during algorithm execution, which is still in line with the tightened-rope observation. The forward graph corresponding to Fig. 4(a) is shown in Fig. 7, where the letter "L" denotes the left group and "R" denotes the right group. We can see that the number of vertices is dramatically reduced in contrast to Fig. 4(b).

We also need to slightly modify the Dijkstra algorithm. Consider Fig. 6 as an example. Assume that the shortest trajectory to vertex LEFT_2 has been discovered through a tangent vector ending at point a in LEFT_2. According to the tightened-rope observation, we can only use forward arcs subtending an angle less than π in reaching other vertices. In order to reach a forward tangent line segment starting from any point in LEFT_2, say c, the arc from a inevitably crosses PU region-1 and is thus invalid. This means that vertex LEFT_2 has become a dead end with no reachable vertices in the forward direction. If this happens, we set the path cost to LEFT_2 as infinity and label it unvisited. This modification makes it possible to reach LEFT_2 via an alternative path and then go further to its neighboring vertices in the forward direction. For example, a new shortest path may be discovered later to LEFT_2 through point b with a valid forward arc to point c which starts another forward tangent line segment.

The computational complexity for employing the Dijkstra algorithm in the suboptimal mode is significantly reduced to $O(N_p^2)$, as there are totally $2N_p + 2$ vertices in the forward graph. But due to procedures taken in Step 1 in Section IV-B, the complexity is maintained at $O(N_p^3)$. This huge advantage is achieved at the expense of possibly suboptimal paths. Consider Fig. 8 as an example, assume that the shortest trajectory has been discovered to vertex LEFT_1 via point a. Then the algorithm will update the edge cost from LEFT_1 to LEFT_2 as $|\widehat{ac}| + |\overrightarrow{cd}|$. What if the length difference between the trajectory from S to vertex LEFT_1 via a and that via b is smaller than $|\widehat{ab}|$? Then the shorter trajectory from S via vertex LEFT_1 to vertex LEFT_2 should traverse the tangent vector ending at b instead of at a. It is worth noting that although such suboptimal trajectories may exist in theory, we rarely observe them in our simulations.

D. Discussion

Now we discuss how to adapt TIGHT to deal with some issues that have been purposefully overlooked so far.

1) Different Transmission Ranges of PUs and SUs: In our network model, the transmission ranges of PUs and SUs are assumed equal. This is for the ease of simulation and representation. However, the application of TIGHT is not constrained by this condition. In cases where different PUs and SUs have different transmission ranges, the source SU can still utilize the proposed scheme to obtain the shortest trajectory as long as the source SU is able to obtain the location information as well as the transmission range of these PUs. Then intermediate SUs still forward the packet within their own transmission ranges based on the embedded trajectory.

2) Impact of topography: In real deployment, the surface might not always be even. It affects the planar graph we build in such a way that the transmission range of PUs is not in the shape of a circle. Thus, considering the level error in this case, we would have a smaller PU region compared to that in the ideal case (the surface is always even). So, in the ideal case, more SUs are considered as ineligible for forwarding the packets due to the possible interference with PUs' communication. Therefore, for the real deployment, it is desirable to model the PU ranges more precisely. It should be noted that this refined modeling requires the knowledge of the topography such as elevation. Depending on the the specific case, we can represent the PU region in the shape of circle, oval, hexagon or other shapes. As long as these shapes can be geometrically

represented, the proposed protocol can still be adapted for deployment. Also, it is worth noting that the original scheme dedicated to the ideal topography can still work in the real topography. Recall that the trajectory determined in the protocol only serves as a guideline for the intermediate forwarding nodes to follow. When considering the next hop candidates, the eligibility of forwarding is still determined by spectrum sensing the environment to make sure no PUs nearby are transmitting. Thus, in real scenario, even we model the PU ranges as perfect circles, the forwarding process launched by secondary users might still take place inside the circles since no PU activity is observed and this is in fact caused by the unevenness of the surface so that the modeled PU range is larger than it is in reality. Admittedly, The performance of our scheme in this case will degrade, leading to possible longer routing paths. However, the impact is limited and will not be a major concern.

3) Node Mobility: Both PUs and SUs may move after a packet leaves S for D. Since TIGHT does not tie the trajectory of every packet to any specific intermediate SU, it is as insensitive to intermediate node mobility as GPSR [10]. The mobility of D can also be easily handled by letting D's neighbors track its movement and then geographically forwarding to D any packet destined to its original destination. S can periodically either get the location update directly from D or query D's location from the location service [12] in case D may have moved far from its previous location S knows.

TIGHT is also highly amenable to PU mobility. Specifically, its greedy mode is nearly immune to PU mobility due to its reliance on local spectrum sensing at every packet forwarder. In contrast, the optimal/suboptimal trajectory computed by S in the optimal/suboptimal mode may be invalid or suboptimal because the PU regions may change due to PU mobility. A simple solution is to allow every intermediate SU to perform local spectrum sensing and modify the trajectory if needed.

4) Computation Complexity: The basic version of TIGHT computes the trajectory in the optimal/suboptimal mode on a per-packet basis. This per-packet computational overhead may be undesirable and can be reduced through batch processing. In particular, S can derive a single trajectory for every α packets, where α is a system parameter. Such batch processing requires S to well predict the locations of the PU regions during the transmission of the α packets involved. This requirement is not so daunting as it seems. In many CR-MANETs we target, the locations of the PUs such as commanding nodes may not be purely random: every PU may move strictly along preplanned physical paths or periodically update its current location and velocity for the next duration with the location service [12]. S can then determine an appropriate trajectory for every α packets based on predictable PU locations. How to measure the quality of a candidate trajectory is very complicated and related to many factors such as α, the mobility pattern of the PUs, and the acceptable overhead for checking with the location service [12]. The further investigation of this issue is left as future work.

5) Insufficient Forwarding Node Density: Geographic routing protocols such as GPSR [10] assume sufficiently high node density. When TIGHT routes a packet along the perimeter of every encountered PU region in the greedy mode or along every trajectory segment close to some PU region, the effective forwarding node density is reduced because the PUs and the SUs inside the PU regions are not allowed to participate in packet forwarding. We can mitigate this issue by simply enlarging the definition of every simple PU region to a circle of radius $2R + \beta$, where β is a system parameter no larger than R. The logic behind this solution can be explained as follows: enlarging the PU region enables the trajectory to deviate from the true original PU region. Then along this trajectory, the intersection of forwarding nodes' region and the neighboring PU regions is shrunk. Thus, compared to the original trajectory, the modified trajectory has more effective forwarding nodes along it since these nodes' transmission will not affect PUs' transmission.

6) Inferior/Infeasible Primary Trajectory: If the trajectory over the primary channel discovered in the optimal/suboptimal mode is more than $R_{\mathrm{p}}/R_{\mathrm{s}}$ times as long as the vector $\overrightarrow{SD}$, it is very likely better to directly route the packet using GPSR towards D using the secondary channel only. This situation can be accommodated by letting S compare the primary trajectory with $\overrightarrow{SD}$ before sending the packet. A better solution may allow packet forwarding over a hybrid trajectory with some segments associated with the primary channel and the others with the secondary channel. This latter solution can also deal with the case that S, D, or both are in some PU regions. How to compute an optimal hybrid trajectory can be rather involved and is beyond the scope of this paper.

7) Multi-Hop PU Communications: We have assumed that every PU only communicates with the nodes within its transmission range when defining PU regions in Section III. This assumption can be relaxed by letting the PUs periodically update the locations of their own and also potential communication partners for the next period. S can then figure out the PU regions which may be non-circular, and the optimal/suboptimal mode of TIGHT can still apply without any modification.

8) Routing Overhead: Every packet forwarded with the optimal/suboptimal node carries the remaining trajectory to traverse. This routing overhead can be reduced by the packet carrying a partial trajectory from S to an intermediate location where the corresponding node computes another partial trajectory towards another intermediate location or D. Obviously, the routing overhead is reduced at the expense of additional computational overhead at intermediate nodes.

V. Performance Evaluation

In this section, we evaluate TIGHT using simulations. There are no publicly available implementations of existing CR-MANET routing protocols [4]–[6], [8], [9]. Also, many key implementation details are not clear enough for us to truthfully realize the functionalities they provide. In addition, as argued in Section I, the great benefits of TIGHT over them are inherited from the well known advantages of GPSR over conventional reactive MANET routing protocols [10]. So we chose to compare TIGHT with GPSR which sends packets over the secondary channel only.

The simulation code was written in C++. Initially, we tried the ns-3 simulator and found that TIGHT has similar packet delivery ratio (PDR) to that of GPSR in small-scale CR-MANETs.

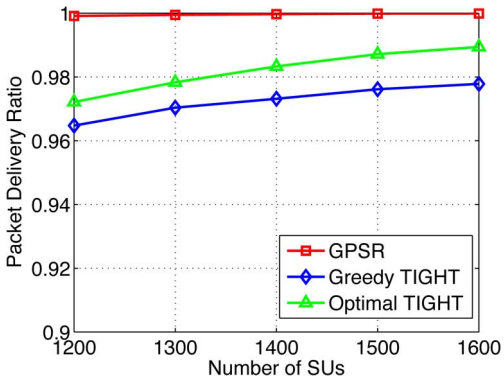

Fig. 9. Impact of different numbers of SUs on packet delivery ratio.

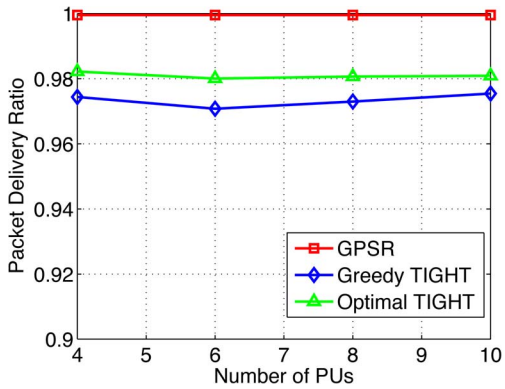

Fig. 10. Impact of different numbers of PUs on packet delivery ratio.

In terms of end-to-end latency, TIGHT only slightly outperforms GPSR due to the small network scale. Since ns-3 is known to have a poor performance in simulating large-scale MANETs, we opted for C++ simulations to see the performance of TIGHT in large-scale networks. The fidelity of C++ simulations for TIGHT and GPSR is still very high. Specifically, the routing of every packet in both TIGHT and GPSR depends on the spontaneous network topology only. Therefore, we can simply evaluate the performance of TIGHT and GPSR under node mobility by sending packets over a number of randomly generated network topologies. Such a simplification cannot be made for reactive routing protocols which bind a route to specific nodes and are thus sensitive to node mobility.

The default simulation settings are as follows. We simulate an area of 4,000 $\times$ 3,000 m^2 with 6 PUs and 1,334 SUs. The transmission range of PUs and SUs is 250 m. We run the simulation 8,000 times, and the PUs and SUs are uniformly distributed at random with a different random seed in each simulation run. We fix 30 pairs of source and destination PUs with random relative locations in each simulation run. TIGHT is used only when the source and destination SUs are both outside of any PU region; otherwise, GPSR is used to send packets over the secondary channel only. Each source SU sends one packet of 512 bytes to its destination SU in each simulation run. Therefore, totally 8,000 packets are sent between each source-destination SU pair, and each data point in subsequent figures represents the average for 240 thousand packets (unless stated otherwise).

As discussed in Section IV-C, we performed simulations on hundreds of randomly generated topologies and there are no differences in the two trajectories obtained using the optimal TIGHT and sub-optimal TIGHT, so for the evaluation below, we use optimal TIGHT to compare with GPSR to demonstrate the effectiveness.

A. *Packet Delivery Ratio (PDR)*

In this section, we show that TIGHT and GPSR have comparable PDRs under various settings.

Fig. 9 compares the PDRs of TIGHT and GPSR under a varying number of SUs. We can see that the PDRs of both TIGHT and GPSR are sufficiently high and increase with the number of SUs. This result is anticipated, as TIGHT and GPSR both require sufficiently high node density to avoid having too many communication voids: the more SUs, the higher the node density, and the fewer packets dropped for lack of a valid next hop towards the destination. In addition, GPSR has slightly better PDR performance than TIGHT. The main reason is that the SUs inside each PU region cannot participate in packet forwarding in TIGHT, which in effect decreases the eligible forwarding node density around the PU regions.

Fig. 10 compares their PDRs when the number of PUs varies. The PDR of GPSR is not affected because GPSR sends packets over the secondary channel only. One may expect the PDR of TIGHT to drop when the number of PUs increases, as a packet may on average take a longer trajectory around more PU regions. We can see that this is not necessarily true. The reason is that the more PUs, the more source/destination SUs falling into PU regions, and the more packets routed over the secondary channel using GPSR. So the PDR of TIGHT does not necessarily decrease with an increasing number of PUs.

The PDR of TIGHT is expected to be improved if we adopt the strategy mentioned in Section IV-D-5. Specifically, every source SU purposefully enlarges every simple PU region size from $2R$ to $2R + \beta$ when computing the per-packet trajectory. In doing so, the density of available forwarding nodes along the modified trajectory can be effectively increased. The impact of β on the PDR of TIGHT in the optimal mode is depicted in Fig. 11, where the PDR for the packets TIGHT sent over the primary channel is also shown. The PDR enhancement is as expected. We observe the similar enhancement in the greedy mode and omit the results here for lack of space. This enhancement comes with two drawbacks. First, the average trajectory length slightly increases from 11.4 hops for $\beta = 0$ to 11.64 hops for $\beta = 200$ m, which may slightly increase the average end-to-end latency. Second, a source SU may not be able to find a valid trajectory traversing enlarged PU regions. Specifically, we observe that the fractions of packets TIGHT sent over the primary channel are 58.6%–53.9%–49.3%–44.9%–40.6% for $\beta = 0|50|100|150|200$ m, respectively. Thus the usage of the primary channel decreases as β increases, which may also increase the end-to-end latency because the primary channel has a higher transmission rate. The further study about the trade-off between PDR and end-to-end latency is underway.

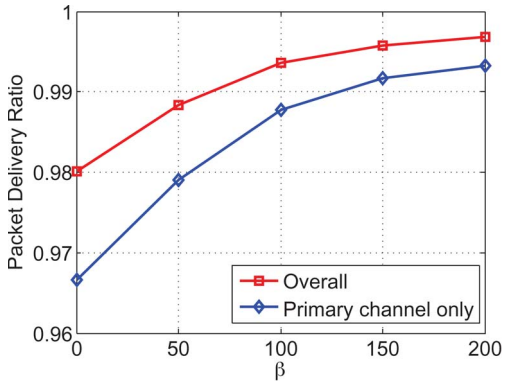

Fig. 11. PDR enhancement of TIGHT.

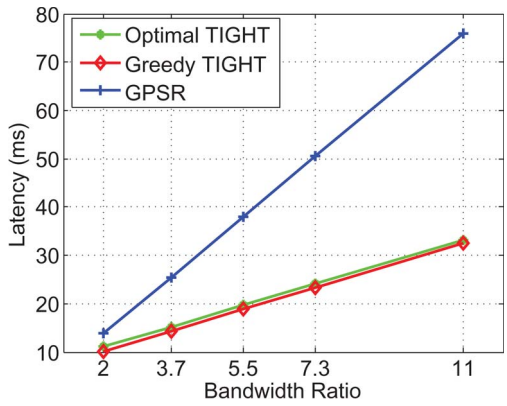

Fig. 12. End-to-end latency.

B. End-to-End Latency

We model the end-to-end latency of TIGHT and GPSR as $k \times (\text{packet\_size}/\text{bandwidth}) \times \#_{\text{hops}}$, where k is a constant system parameter. k is set as the same value 1 for both TIGHT and GPSR in our simulation to make a fair comparison. In real scenarios, k might vary. So we add this parameter in the model for completeness. The value of k does not affect our simulation and neither our conclusions in the scheme comparison. According to this model, the latency is also proportionate to $\#_{\text{hops}}$ and inverse proportionate to bandwidth. If a packet is sent over the primary channel, R_{p} is used as the bandwidth; otherwise, R_{s} is used. The packet_size accounts for 1,024 bytes for the data content and the routing information added by TIGHT or GPSR. Specifically, all schemes add the destination location to a packet, which is assumed to be four bytes long with two bytes for x-coordinate and y-coordinate each. GPSR also needs to record the perimeter mode entry point location, etc. The greedy mode in TIGHT has to append the routing direction, PU avoidance entry point location, etc., in the packet header when entering PU avoidance phase, which corresponds to less than five bytes in total. Also, the greedy mode in TIGHT needs to record similar information like GPSR when perimeter mode is invoked. In contrast, the optimal mode of TIGHT adds the trajectory to every packet. Each trajectory consists of line segments and circular arc segments, and each line or arc segment can be represented by its two end locations, corresponding to eight bytes in total.

Fig. 12 shows the impact of the bandwidth ratio $R_{\text{p}}/R_{\text{s}}$ on the end-to-end latency of TIGHT and GPSR. As we can see, the greedy and optimal modes of TIGHT always outperform GPSR due to the aggressive use of the primary channel, and the gap between them increases with $R_{\text{p}}/R_{\text{s}}$. Such results highlight the advantages of cognitive communications. In addition, the greedy mode incurs slightly shorter end-to-end latency than the optimal mode mainly due to the longer routing header in the optimal mode, and this advantage comes with the cost of a lower PDR as shown in Figs. 9 and 10. We have also observed that the latency gap between them becomes negligible if the strategy in Section IV-D-8 is adopted to let intermediate nodes remove traversed line and arc segments. Moreover, the optimal mode has much better latency performance in some scenarios as shown below.

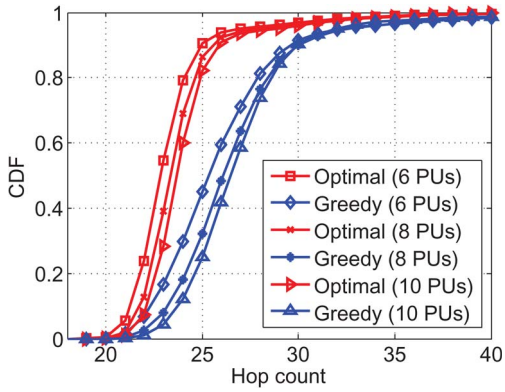

Fig. 13. Optimal vs. Greedy: Case 1.

C. Greedy or Optimal?

Our previous results show that the optimal mode of TIGHT has a slightly higher PDR than the greedy mode with comparable end-to-end latency. In this section, we focus on comparing the two modes by measuring their respective path length in the number of hops in some special cases.

1) Case 1: Distant Source-Destination SUs: We conjecture that when the source and destination SUs are far apart with many PUs blocking the straight-line segment between them, every packet will traverse along a much shorter trajectory computed in the optimal mode in contrast to the greedy mode. To validate our conjecture, we fix a source SU and the corresponding destination PU at the opposite corners of the $4,000 \times 3,000$ m^2 rectangular region. Other SUs are still uniformly distributed at random. Fig. 13 shows the cumulative distributed function (CDF) functions of the path length for both modes after 8,000 simulation runs. As we can see, the optimal mode statistically finds shorter forwarding paths than the greedy mode. For example, when there are 10 PUs, the mean path lengths are 24.67 and 27.43 hops for the optimal and greedy modes, respectively. Meanwhile, the average path lengths for both modes increase with the number of PUs.

We also evaluate the stretch factor of TIGHT using simulation. The stretch factor is the ratio of the actual path length to the shortest path length. The result is shown in Fig. 14. We observe for the majority (above 90%) of routes derived in optimal mode, the stretch factor is in the range from 1 to 1.1,

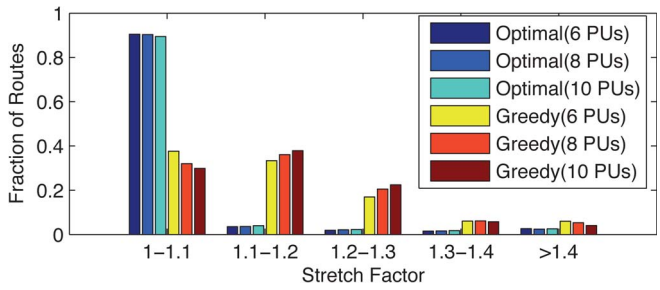

Fig. 14. Stretch factor distribution.

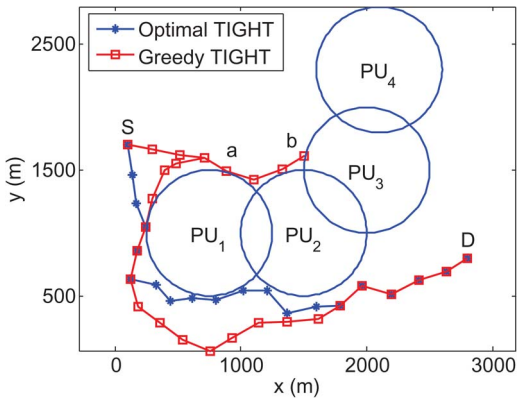

Fig. 15. Optimal vs. Greedy: Case 2.

indicating the obtained routes are the true shortest paths or close to them. However, for the routes obtained in greedy mode, the distribution of the stretch factor is more spread out and flatter. Specifically, we notice the fraction of packets for the stretch factor being 1 to 1.1 and 1.1 to 1.2 are close. Even when stretch factor increases to 1.2, there is still a generally large proportion (above 20%) of routes falling into this category.

2) Case 2: Complicated Topology: We also compare the two modes in a special topology in Fig. 15, where the actual geographic forwarding paths for both modes are shown and marked with dots and squares, respectively. In the greedy mode, the packet first reaches point b through a and thus enters the PU avoidance phase. Since the counterclockwise direction is decided, the packet is forwarded back to a and continues on the line with square marks. The optimal mode, however, manages to first identify the shortest trajectory and thus avoids taking a detour towards the destination D. In this example, the path length in the optimal and greedy modes is 19 and 31 hops, respectively. Correspondingly, the stretch factor is 1 and 1.63, respectively. The advantage of the optimal mode over the greedy mode can be more profound in more complicated network topology.

3) Case 3: PU Activities: We also compare the two modes when the PU activity pattern varies in Fig. 16. The PU activity factor refers to the possibility for PUs to be at ON state (PUs keep transmitting). We observe that for the greedy mode, the performance is affected by the PU activity factor in terms of average hop count while for the optimal mode, the performance is always stable. Our results confirm that the greedy mode leads to shorter forwarding paths when the PUs are less active, while the optimal mode excels when the PUs are very active.

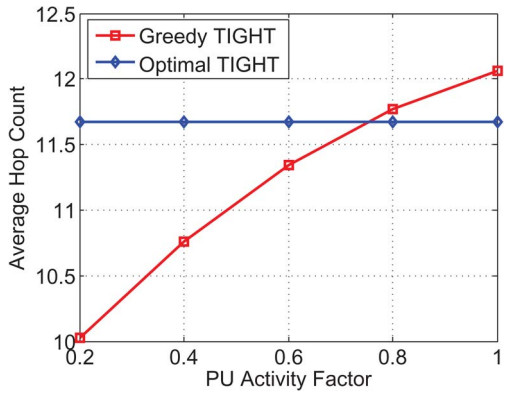

Fig. 16. PU activity.

This conclusion, however, does not contradict with the term "optimal" for the optimal mode. The reason is that when PUs are off, the trajectory generated by S is not updated in a timely fashion. S will still generate the trajectory assuming PUs are all transmitting. Thus, the trajectory is always the same providing S does not get the updated activity information of the PUs. If there is a mechanism that updates S about PUs' activity, then the optimal mode can still generate the shortest path. Without loss of generality, in the simulation, we don't assume the existence of this mechanism. Another remedy is provided in Section IV-D-3. Every intermediate SU can perform local spectrum sensing and modify the trajectory if needed. This solution does not rely on the previously mentioned mechanism, but can increase the power consumption of the intermediate SUs due to the involved computation. Therefore, in the simulation, we haven't provided the support of this as well. By showcasing the result in Fig. 16, the difference of these two schemes is hopefully better understood.

VI. Conclusion

This paper presents TIGHT, a geographic routing protocol for cognitive radio mobile ad-hoc networks. TIGHT offers three routing modes to enable secondary users to fully explore spectrum opportunities over the primary channel without imposing interference on primary users. The greedy mode does not require the knowledge of the global primary user location information, and can route around the PU region when all valid candidate forwarding nodes are outside of the PU region. The optimal mode assumes the global primary user location information is available, thus the source can derive a shortest trajectory towards the destination before sending out packets and intermediate forwarding nodes can also refer to the embedded trajectory information to choose the next hop candidate. The suboptimal mode lowers the computation complexity of the optimal mode with the trade-off of possibly using suboptimal trajectory. The effectiveness of the protocol has been validated through extensive simulations.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments and helpful advice.

REFERENCES

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Comput. Netw.*, vol. 50, no. 13, pp. 2127–2159, May 2006.

[2] B. Wang and K. J. R. Liu, "Advances in cognitive radio networks: A survey," *IEEE J. Sel. Areas Commun.*, vol. 5, no. 1, pp. 5–23, Feb. 2011.

[3] M. Cesana, F. Cuomo, and E. Ekici, "Routing in cognitive radio networks: Challenges and solutions," *Ad Hoc Netw.*, vol. 9, no. 3, pp. 228–248, May 2011.

[4] G. Cheng, W. Liu, Y. Li, and W. Cheng, "Spectrum aware on-demand routing in cognitive radio networks," in *Proc. IEEE DYSPAN*, Dublin, Ireland, Apr. 2007, pp. 571–574.

[5] A. Sampath, L. Yang, L. Cao, H. Zheng, and B. Zhao, "High throughput spectrum-aware routing for cognitive radio networks," in *Proc. CROWNCOM*, Singapore, May 2008, pp. 1–6.

[6] G.-M. Zhu, I. Akyildiz, and G.-S. Kuo, "STOD-RP: A spectrum-tree based on-emand routing protocol for multi-hop cognitive radio networks," in *Proc. IEEE GLOBECOM*, Nov. 2008, pp. 1–5.

[7] K. Chowdhury and M. Felice, "SEARCH: A routing protocol for mobile cognitive radio ad-hoc networks," *Comput. Commun.*, vol. 32, no. 18, pp. 1983–1997, Dec. 2009.

[8] K. Chowdhury and I. Akyildiz, "CRP: A routing protocol for cognitive radio *Ad Hoc* networks," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 4, pp. 794–804, Apr. 2011.

[9] A. Cacciapuoti, M. Caleffi, and L. Paura, "Reactive routing for mobile cognitive radio *ad hoc* networks," *Ad Hoc Netw.*, vol. 10, no. 5, pp. 803–815, Jul. 2012.

[10] B. Karp and H. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. ACM MOBICOM*, Boston, MA, USA, Aug. 2000, pp. 243–254.

[11] D. Niculescu and B. Nath, "Trajectory based forwarding and its applications," in *Proc. ACM MobiCom*, San Diego, CA, USA, Sep. 2003, pp. 260–272.

[12] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris, "A scalable location service for geographic *ad hoc* routing," in *Proc. MobiCom*, Boston, MA, USA, Aug. 2000, pp. 120–130.

[13] W. Kieß, H. Füßler, J. Widmer, and M. Mauve, "Hierarchical location service for mobile ad-hoc networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 8, no. 4, pp. 47–58, Oct. 2004.

[14] H. Celebi and H. Arslan, "Utilization of location information in cognitive wireless networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 4, pp. 6–13, Aug. 2007.

[15] C. Perkins, E. Belding-Royer, and S. Das, *Ad Hoc* On-Demand Distance Vector (AODV) Routing, RFC 3561, Jul. 2003.

[16] D. Chen and P. Varshney, "A survey of void handling techniques for geographic routing in wireless networks," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 1, pp. 50–67, 2007.

[17] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in *Ad Hoc* wireless networks," in *Proc. 3rd Int. Workshop DIALM*, Aug. 1999, pp. 48–55.

[18] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutić, "Routing with guaranteed delivery in *ad hoc* wireless networks," *Wireless Netw.*, vol. 7, no. 6, pp. 609–616, Nov. 2001.

[19] D. B. Johnson and D. A. Maltz, "Dynamic source routing in *ad hoc* wireless networks," *Mobile Comput.*, vol. 353, pp. 153–181, 1996.

[20] D. Cabric, A. Tkachenko, and R. W. Brodersen, "Experimental study of spectrum sensing based on energy detection and network cooperation," in *Proc. 1st Int. Workshop TAPAS*, Aug. 2006, p. 12.

[21] H. Kim and K. G. Shin, "In-band spectrum sensing in cognitive radio networks: Energy detection or feature detection?" in *Proc. MobiCom*, Sep. 2008, pp. 14–25.

[22] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 1, pp. 116–130, 2009.

[23] S. Haykin, D. J. Thomson, and J. H. Reed, "Spectrum sensing for cognitive radio," *Proc. IEEE*, vol. 97, no. 5, pp. 849–877, May 2009.

[24] L. Khaled and Z. Wei, "Cooperative communications for cognitive radio networks," *Proc. IEEE*, vol. 97, no. 5, pp. 878–893, May 2009.

Xiaocong Jin received the B.E. degree in information engineering and the M.S. degree in signal and information processing from Shanghai Jiao Tong University, Shanghai, China, in 2009 and 2012, respectively, and the M.S. degree in information, production, and systems engineering from Waseda University, Tokyo, Japan, in 2010. He is currently working toward the Ph.D. degree at the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ, USA.

His main research interests include security and privacy of network and distributed systems, wireless networking, and mobile computing.

Rui Zhang (M'13) received the B.E. degree in communication engineering and the M.E. degree in communication and information systems from Huazhong University of Science and Technology, Wuhan, China, in 2001 and 2005, respectively, and the Ph.D. degree in electrical engineering from Arizona State University, Tempe, AZ, USA, in 2013.

From 2005 to 2007, he was a Software Engineer with the UTStarcom Shenzhen R&D Center. Since July 2013, he has been an Assistant Professor with the Department of Electrical Engineering, University of Hawaii, Honolulu, HI, USA. His main research interests include network and distributed system security, wireless networking, and mobile computing.

Jingchao Sun received the B.E. degree in electronics and information engineering and the M.E. degree in communication and information System from Huazhong University of Science and Technology, Wuhan, China, in 2008 and 2011, respectively. He is currently working toward the Ph.D. degree at the School of Electrical, Computer, and Energy Engineering, Arizona State University, Phoenix, AZ, USA.

His main research interests include security and privacy of network and distributed systems, wireless networking, and mobile computing.

Yanchao Zhang (SM'11) received the B.E. degree in computer science and technology from Nanjing University of Posts and Telecommunications, Nanjing, China, in 1999, the M.E. degree in computer science and technology from Beijing University of Posts and Telecommunications, Beijing, China, in 2002, and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2006.

He is currently an Associate Professor with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ, USA. His main research interests include network and distributed system security, wireless networking, and mobile computing. He is an Editor of IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, and IEEE WIRELESS COMMUNICATIONS. He was a Technical Program Committee Cochair of the Communication and Information System Security Symposium at the 2010 IEEE Global Communications Conference.

Dr. Zhang received the NSF CAREER Award in 2009.